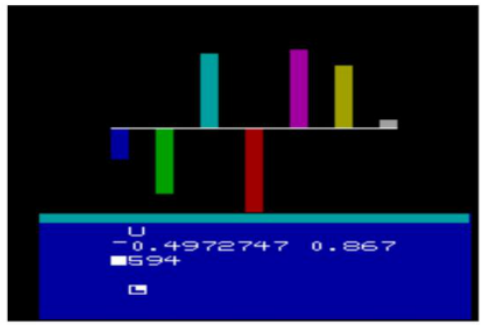


21 APL/S Games for Your Umtech VideoBrain

by James Price

21 APL/S Games and One-Liners for Your Umtech VideoBrain

APL Games for Microcomputers, Volume 1

James Price

Published by James Price, 2025.

While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

21 APL/S GAMES AND ONE-LINERS FOR YOUR UMTECH VIDEOBRAIN

First edition. December 19, 2025.

Copyright © 2025 James Price.

Written by James Price.

Table of Contents

Title Page

Copyright Page

21 APL/S Games and One-Liners for Your Umtech VideoBrain (APL Games for Microcomputers, #1)

Acknowledgments

Dedication

Series Introduction

Introduction to APL/S on the Umtech VideoBrain running on MAME

Slot Machine

Sine Barchart One-Liner

Acey Ducey

Roulette

Gunner

Collatz Bounce

Phi and Fibonacci One-Liners

Pyramid chart

Recursive Factorial Fun

Lunar Lander

Unit Circle

Dicejack

Primes

Data Statistics

Lotto Picker

Estimate Pi

Taylor, Maclaurin, and Madhava-Leibniz Series

Stag Hunt

Moo

Ecte’s Guess

Recursive Coin Change Maker

Calendar Clock

Credits

The APL/S games in this book are collected here:

Sign up for James Price's Mailing List

Copyright © 2025 James Price. All rights reserved.

This publication is an independent, unofficial work and is not authorized, licensed, sponsored, or endorsed by Umtech, Inc., or by any successor, assignee, or holder of intellectual property rights associated with the Umtech VideoBrain computer system. Umtech, Inc. is no longer in operation.

All programs contained herein are original works by the author unless otherwise indicated. Programs are written in APL/S for compatibility with the Umtech VideoBrain. No original Umtech software, firmware, ROM code, source code, or proprietary documentation is reproduced, distributed, or derived. Any trademarks or product names referenced are used solely for identification and compatibility purposes and remain the property of their respective owners. While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

Acknowledgments

Thanks to Sean Riddle for making this all possible by performing the heavy lifting and technical magic needed to resurrect the APL/S ROMs so we can emulate The Computational Language of the Umtech VideoBrain using MAME. Thanks also to the APL/S cartridge donor, those on the MAME team who added the VideoBrain computer system to MAME, and the VideoBrain enthusiast community for keeping this system in their hearts all these years.

Thanks to The Array Cast team of Bob Therriault, Conor Hoekstra, Marshall Lochbaum, Adám Brudzewsky, and Stephen Taylor for having me on the show and encouraging me to push forward with writing APL games books. Thanks also to Marshall Lochbaum for his enthusiasm in documenting APL/S on the APL Wiki, and for hosting the Discord APL Farm “Quad Squad,” a weekly discussion of ACM SIGAPL Quote Quad articles, which lead to the port of the Unit Circle program.

Thanks to Elias Mårtenson for his encouragement and interest in this project and suggestions for improvements to the upcoming print edition.

Dedication

To my beloved tabby cat, Mr. Tibbs.

Thanks for hanging out with me while I was writing these games during late Summer and Autumn of 2025, little buddy.

Series Introduction

An early APL gaming culture left behind a rich treasure trove of artifacts now readily accessible to code archaeologists. Early issues of *Quote Quad* contained APL implementations of Poker, Target, Conway's Game of Life, and Tic Tac Toe. A game called Pico-Fumi is found in the *Proceedings of the Fifth International APL Users' Conference* (May 1973). Gaming historians digging through program listings on MTS systems running on IBM System/360s will uncover a resource management game called Kingdom written in APL. An issue of the I. P. Sharp newsletter called for submissions of WARI (Mancala) playing programs. The submitted entries then fought a series of battles, and the winners were reported in a follow-up issue.

I.P. Sharp newsletters from the late 1970's document a 91 FUNSPACE workspace, and track the evolution of 92 GAMES, which included ELIZA, STARTREK, SPACEWAR, GAMBLE, SNIM, WARGAMES, WORDGAMES, MASTERMIND, QUADGT, WUMPUSHUNT, and a game called "Bridge Hands".

The I.P. Sharp Public Workspaces Catalogue entry for 92 GAMES from 1981 lists "tictactoe, blackjack, showdown poker, slot machine, craps, horserace, supernim, gunfire games, hangman, scramble, and finite-state machine, nim, mastermind, wumpus, and quadgit." It also includes 93 LIFE: "This workspace contains Life, a solitaire game involving a world in which cells live and die according to a few simple rules. The rules have been chosen to make the behaviour of the cell population quite interesting."

The MCM/70's APL library LIB/70 included a "Fun and Games Library" in 1974 consisting of "several games, such as HOR (HORSE, possibly the world's earliest game implemented on a microcomputer), HAN (HANGMAN), GUN, etc." Reviewing the tape, one also finds the source code to games like Moo, Craps, and Blackjack.

Back then, it was common to type in programs from print media. A Lunar Lander game appears in the APL-themed August 1977 issue of *Byte* magazine, immediately following Ken Iverson's article "Understanding APL." Type-in programs were published in

magazines, like Creative Computing and Compute's Gazette, and in books of games written in BASIC by authors like David Ahl and Tim Hartnell. BASIC was built into the microcomputers used in schools, Apple IIs and Commodore PETs, and in homes, Timex/Sinclair 1000s, Commodore 64s, and Atari 800XLs.

David Ahl explained in an interview: "BASIC was the only widely available interactive language. Smalltalk? Lisp? Logo? MUMPS? They didn't stand a chance against BASIC." Steve Wozniak recalls his motivation for writing Integer BASIC for the Apple computer: "There was a book out, 101 BASIC Computer Games, and Bill Gates had written in BASIC. So I said: You've got to have these games; it's going to be the heart of a computer that's worth anything to people..."

But with BASIC everywhere, many critics lamented the harm such a questionable language would inflict on vulnerable programming populations. Early use of BASIC was said to leave one "mentally mutilated beyond hope of regeneration." Did learning such a poorly crafted tool of thought first leave malleable young minds permanently misshapen? To what extent did the use of BASIC represent a lost opportunity to introduce the microcomputer generation to structured programming?

What if, by a twist of fate, instead of BASIC, APL had been the language built into home microcomputers? What if an APL workspace had greeted programmers instead of the BASIC prompt? Like BASIC, APL is interpreted and well-suited to writing games. The Waterloo APL disk for the Commodore SuperPET shipped with a version of MASTERMIND, and the APL/Z80 disk included an implementation of Conway's Game of Life.

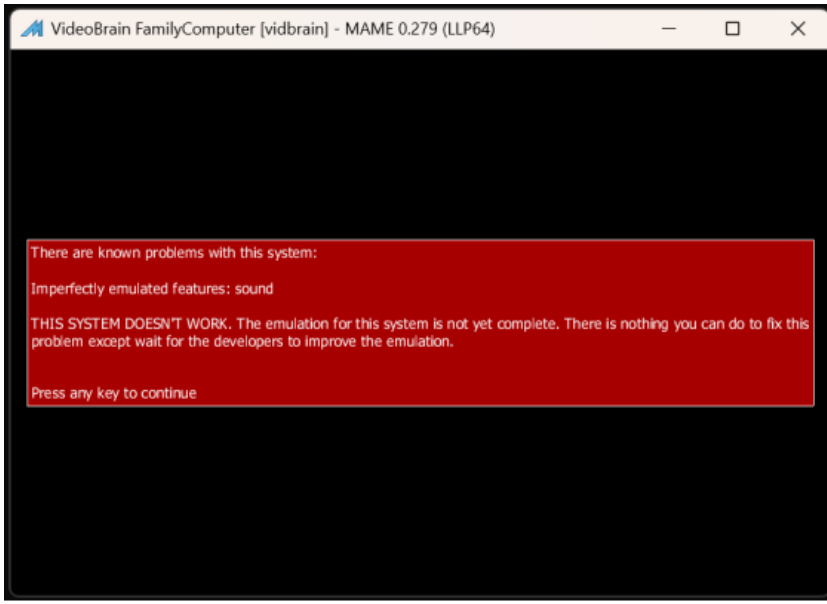
So what might an APL games book have looked like, if APL had been the standard built-in language? These books explore APL in the spirit of learning the language by writing some of those games. This series will examine APL variants running on the Umtech VideoBrain, the Commodore SuperPET, the Commodore 64 and C128, and the MCM/70.

Introduction to APL/S on the Umtech VideoBrain running on MAME

The 1977 Umtech VideoBrain's programming language offering, The Computational Language cartridge, provided users with an interactive APL/S workspace with 920 bytes of free memory. All user programs and data, including variables and arrays, needed to fit into this tiny workspace. Numbers were stored as 4-byte floating points, with program keywords and variable references stored as single byte tokens.

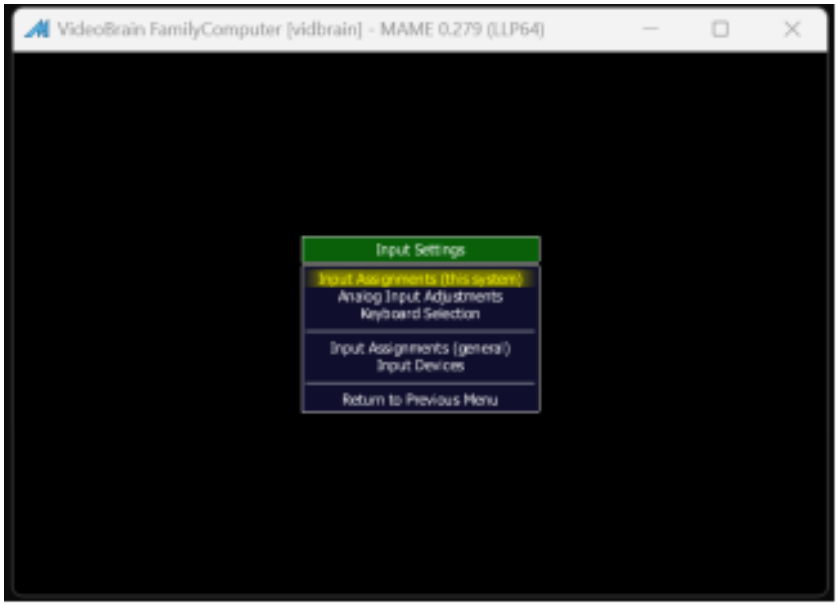
Working within this very small workspace can be a challenge, especially when including visualization techniques provided by the primitive bar graph facility. Most of the games in this book use the bar graph feature in some way. Many of these games use almost every single byte of memory. Part of the fun in writing games for the VideoBrain is to see if you can squeeze in just one more feature, like adding leap year handling code to the clock calendar program. Sometimes a little space can be freed up by leaving initial variable assignments out of the program, and only typing them in before running it, or by shortening text strings used in output. The APL/S language is a good choice for code golfing.

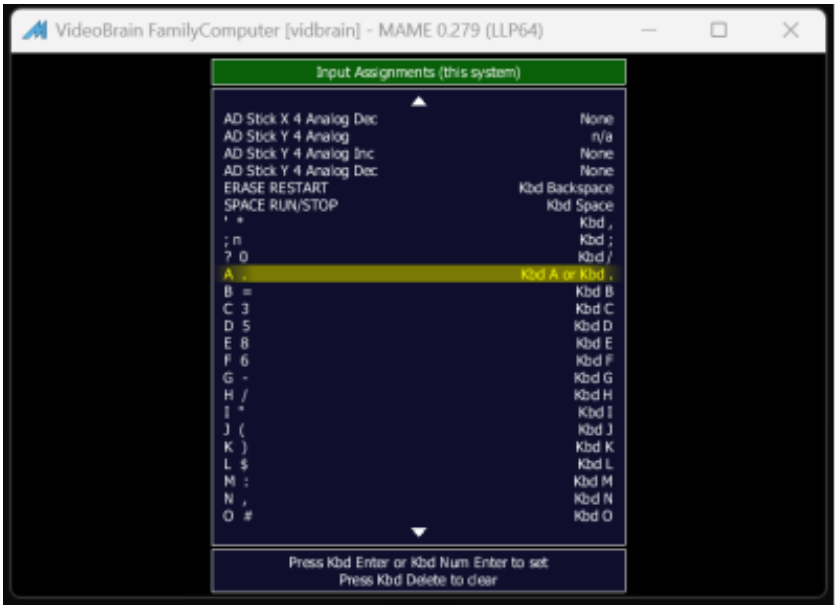
Using MAME makes program development a lot easier, since you may save and load games using the debugger. The debugger can be used to examine memory and the stack. MAME even lets you speed up the system by skipping frames. You can remap keys, and I found the 'A' key needed to be remapped to the '.' key since the real 'A' key was inexplicably showing up as the 'O' key.


VideoBrain FamilyComputer [vidbrain] - MAME 0.279 (LLP64)
There are known problems with this system:
Imperfectly emulated features: sound
THIS SYSTEM DOESN'T WORK. The emulation for this system is not yet complete. There is nothing you can do to fix this problem except wait for the developers to improve the emulation.
Press any key to continue

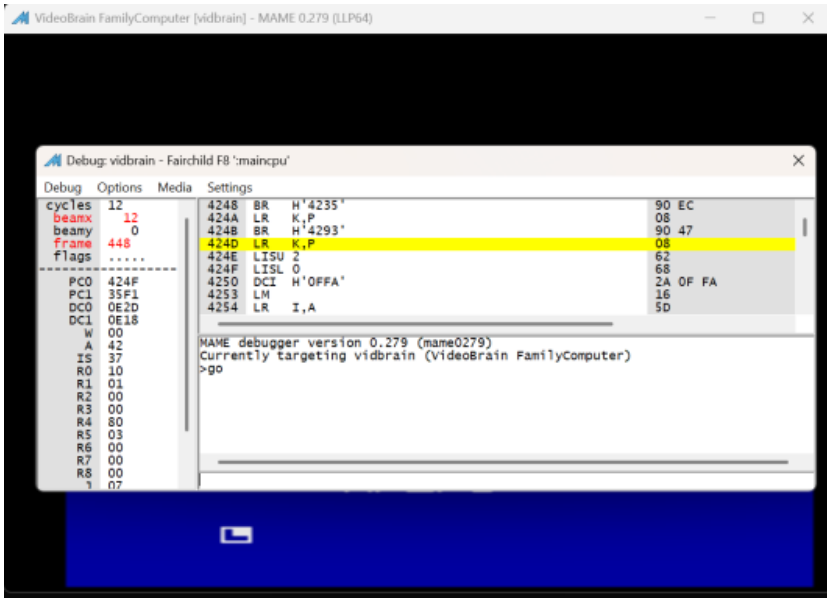
If you also need to remap the ‘A’ key, here’s how: first, when loading The Computational Language cart, hit the TAB key here.

VideoBrain FamilyComputer [vidbrain] - MAME 0.279 (LLP64)
Input Settings
Bookkeeping Info
System Information
Warning Information
Media Image Information
File Manager
Slot Devices
Audio Mixer
Audio Effects
Slider Controls
Video Options
Add To Favorites
About MAME
Start System

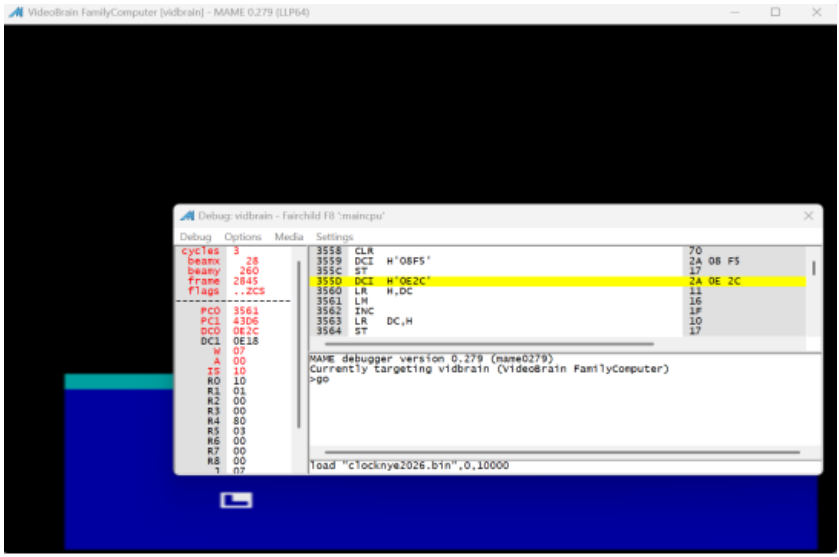

VideoBrain FamilyComputer [vidbrain] - MAME 0.279 (LLP64)
Input Settings
Input Assignments (this system)
Analog Input Adjustments
Keyboard Selection
Input Assignments (general)
Input Devices
Return to Previous Menu

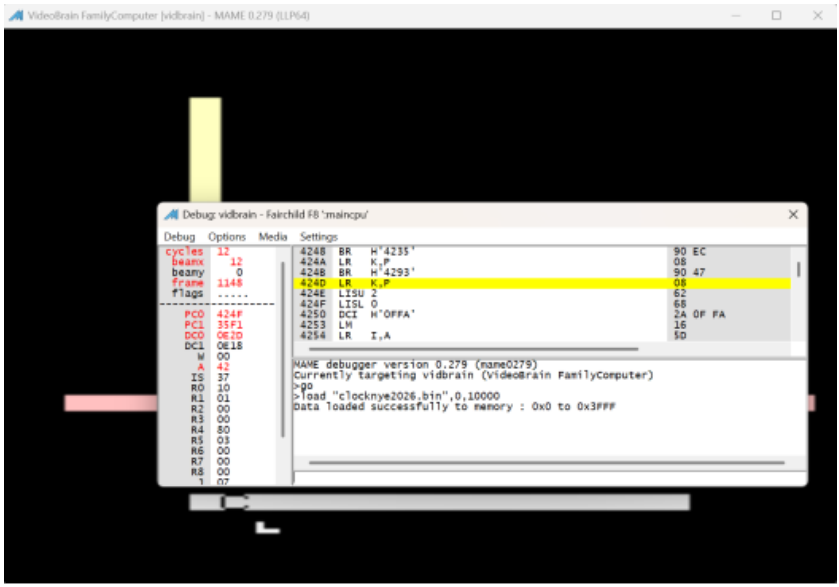

VideoBrain FamilyComputer [vidbrain] - MAME 0.279 (LLP64)
Input Assignments (this system)
AD Stick X 4 Analog Dec None
AD Stick Y 4 Analog n/a
AD Stick Y 4 Analog Inc None
AD Stick Y 4 Analog Dec None
ERASE RESTART Kbd Backspace
SPACE RUN/STOP Kbd Space
' * Kbd ,
; n Kbd ;
? 0 Kbd /
A . Kbd A or Kbd .
B = Kbd B
C 3 Kbd C
D 5 Kbd D
E 8 Kbd E
F 6 Kbd F
G - Kbd G
H / Kbd H
I " Kbd I
J (Kbd J
K) Kbd K
L $ Kbd L
M : Kbd M
N , Kbd N
O # Kbd O
Press Kbd Enter or Kbd Num Enter to set
Press Kbd Delete to clear

Then, go to the Input Settings for the keyboard keys, set the new ‘A’ key, and you’re good to go.



To load games in the debugger, launch MAME with the -debug option (you can also use the -window option to run MAME in a window). When the debugger starts up, type ‘go’ and hit return.

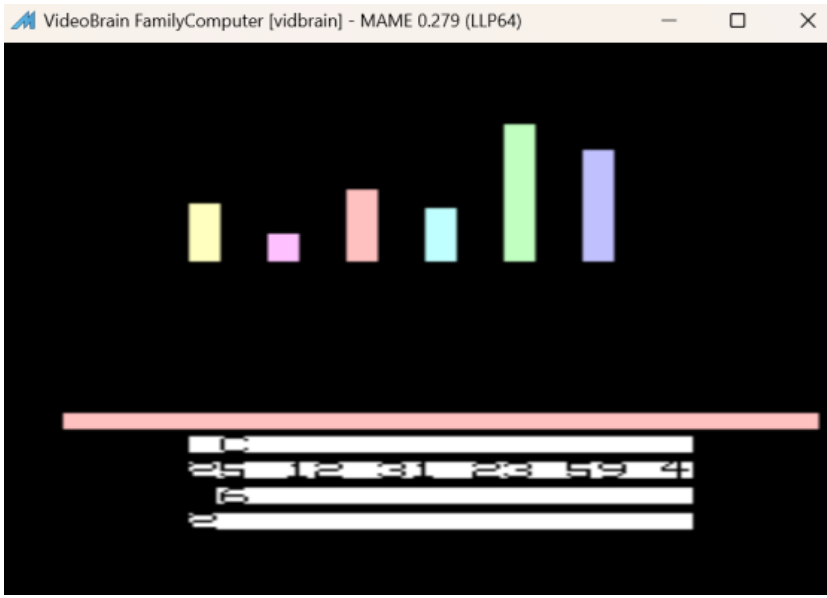




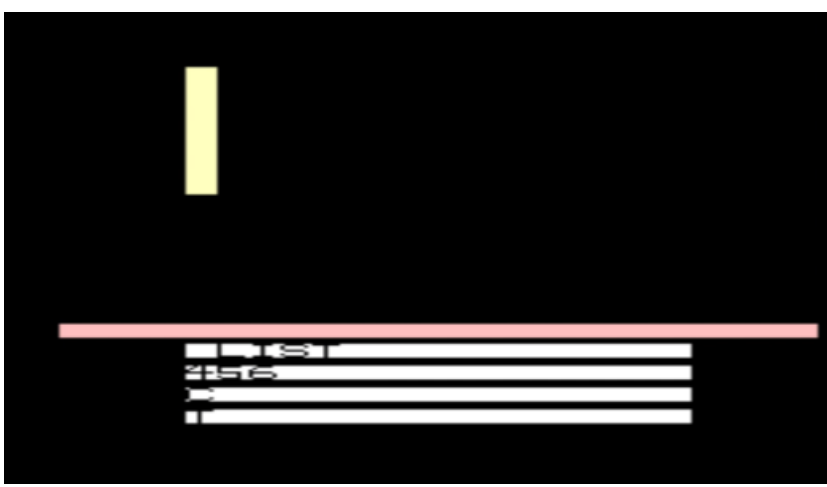
Now load a game by typing “load GAME.bin,0,10000” where GAME is the game name. Games are similarly stored by typing “save GAME.bin,0,10000” in the debugger. Click the debugger closed (it can be re-opened with the “`” key).

VideoBrain FamilyComputer [vidbrain] - MAME 0.279 (LLP64)

The game is now loaded, and ready to run. Hit the F3 key to execute the program.

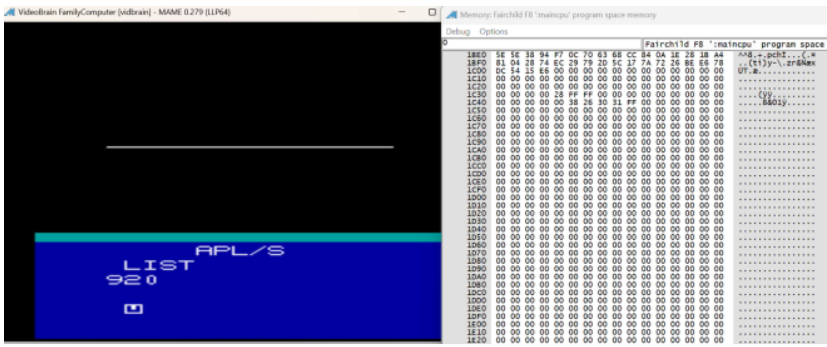


The program starts to run. The screen will look a little different than it usually looks after loading in a program using the debugger.

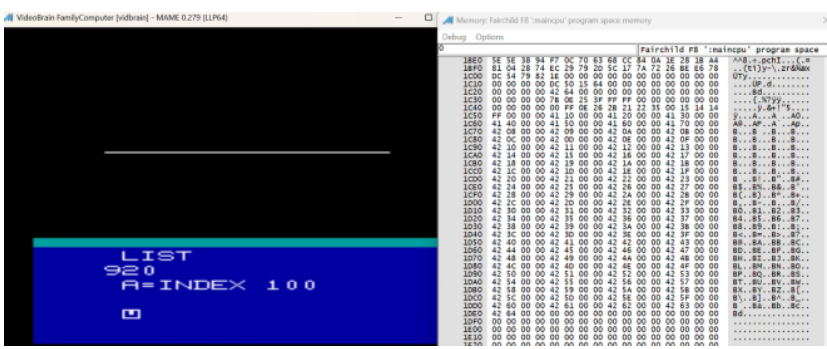


Alternatively, instead of running the program, you can LIST the variables and programs. Hit F1 to use the backspace to remove the program name, ‘C’, and then type LIST and hit F3 to enter the command.

LIST causes the free space remaining in the workspace to be shown, followed by the names of all programs and variables used.



The debugger can easily be used to examine programs and variables in memory. Click on Debug, then select New Memory Window to open a window to view the system memory.



We quickly fill up 400 bytes of the 920 bytes of free memory by creating an array of 100 numbers.

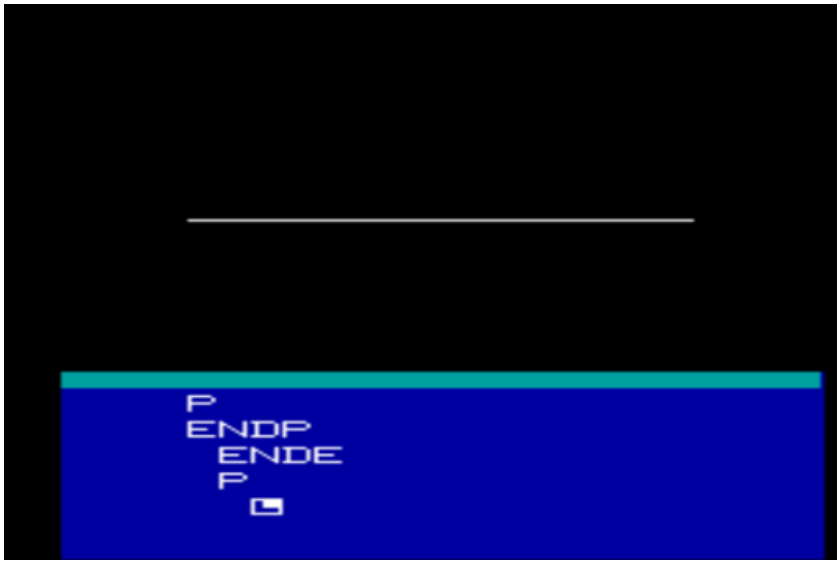
Watching the memory window during program runs can be very entertaining and offers insights into the operation of the system. We’ll examine the stack operation in the Recursive Factorial Fun program chapter.

As far as I can tell, you cannot run the debug mode and use frame skipping at the same time. To speed up the system, run MAME without the debug flag, then increase system speed using frame skipping as usual. The only drawback seems to be that one cannot speed up a game loaded in using the debugger.

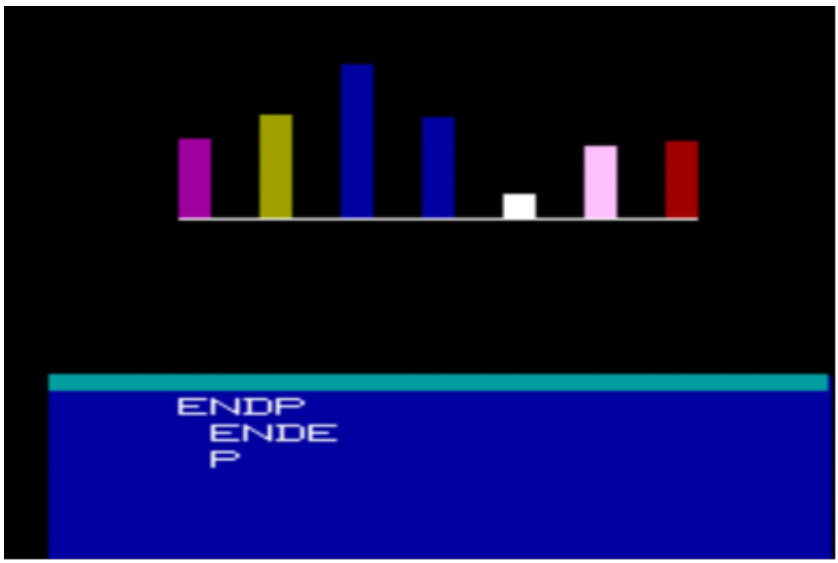
EDIT P
PROG P
BARH RAND (64+A
RRA 7)

To enter in a new program, type EDIT PROGNAME, for example, EDIT P.

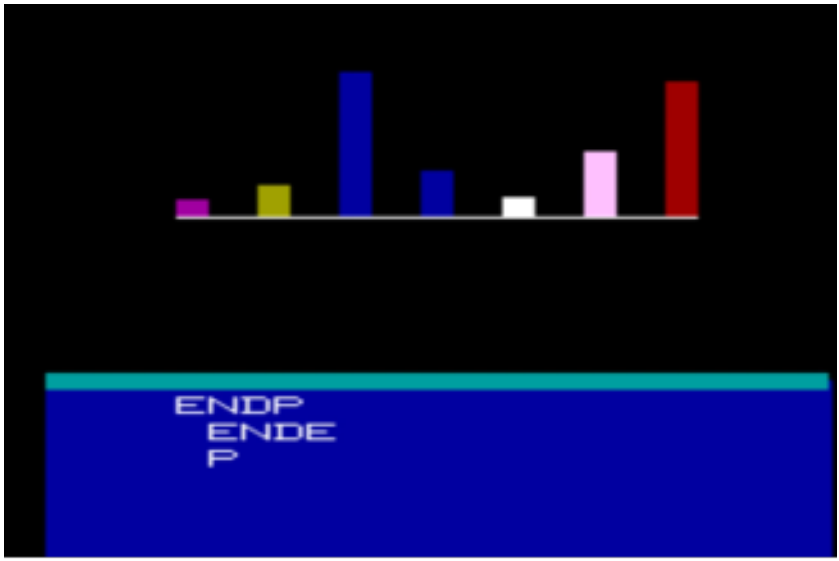
End the program with ENDP.

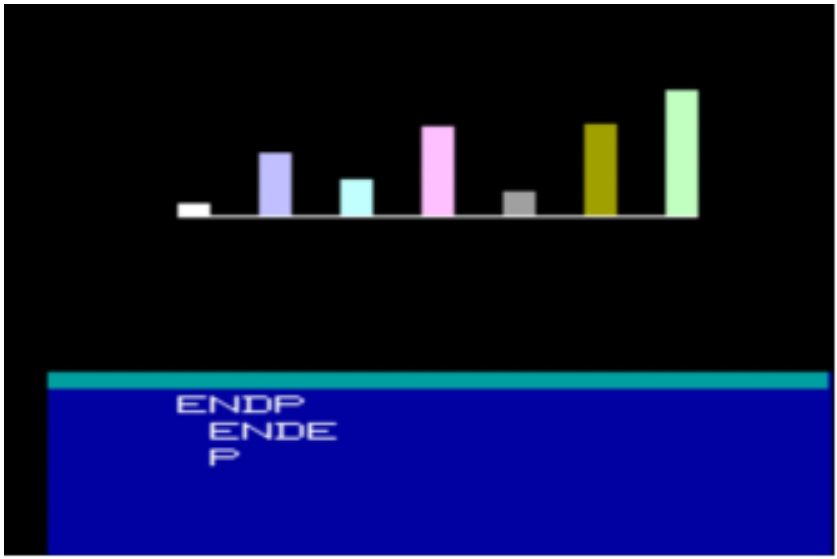


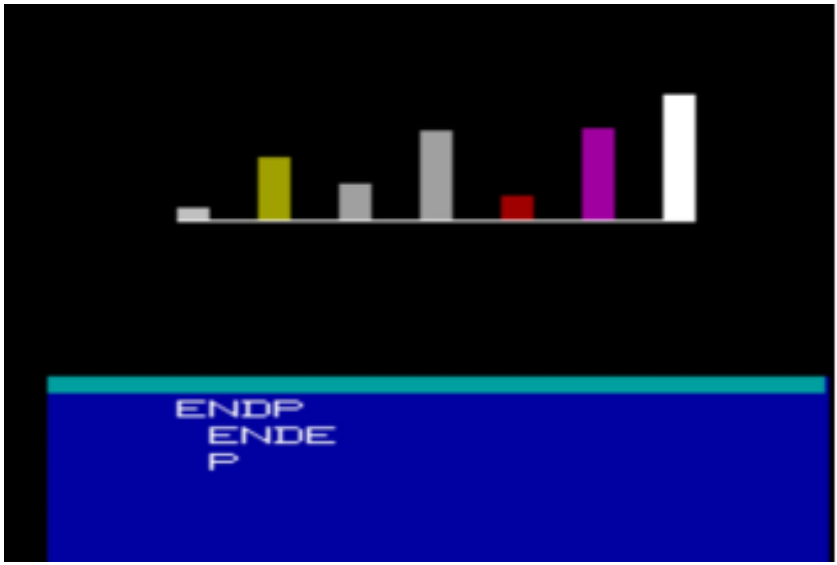
Type ENDE to exit the program editor. Then run the program by typing ‘P’ and hitting F3. (F2 moves the editor back one line, and F4 opens a new line in an existing program.)

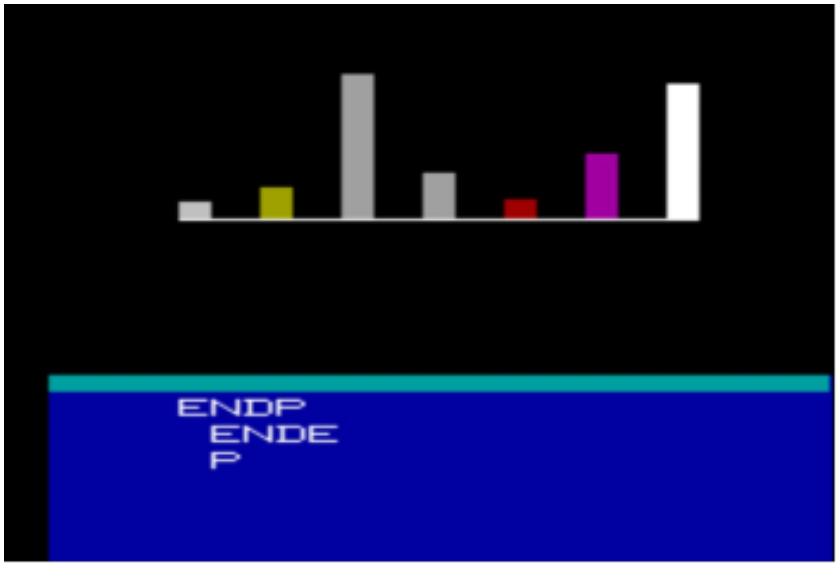


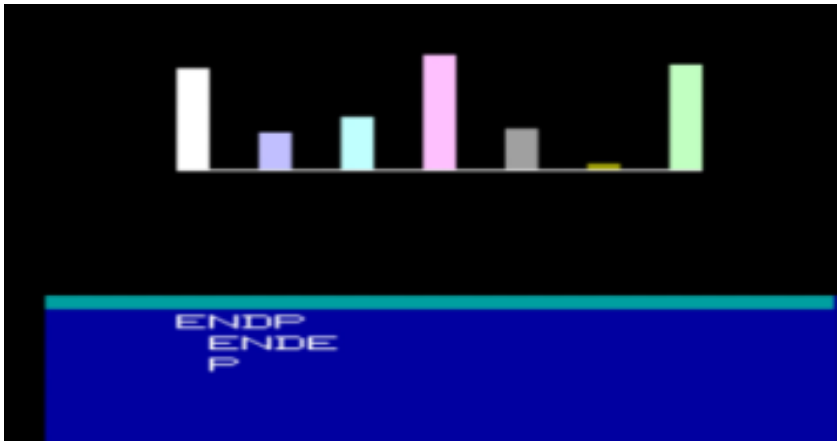
The program generates bar graphs of random height and color, then calls itself recursively.

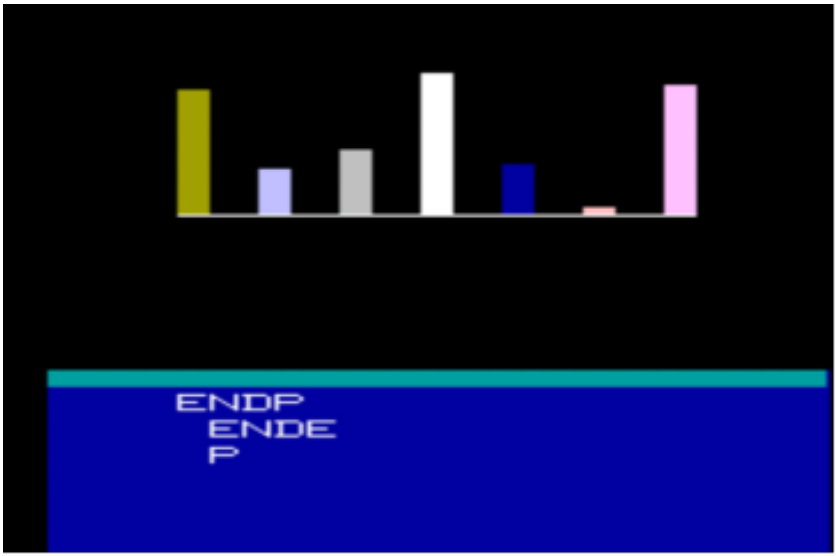


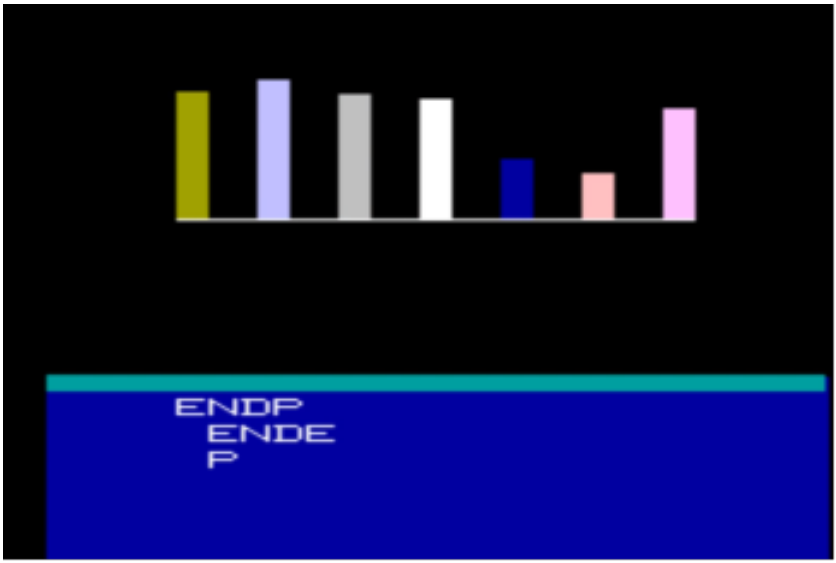

ENDP
ENDE
P

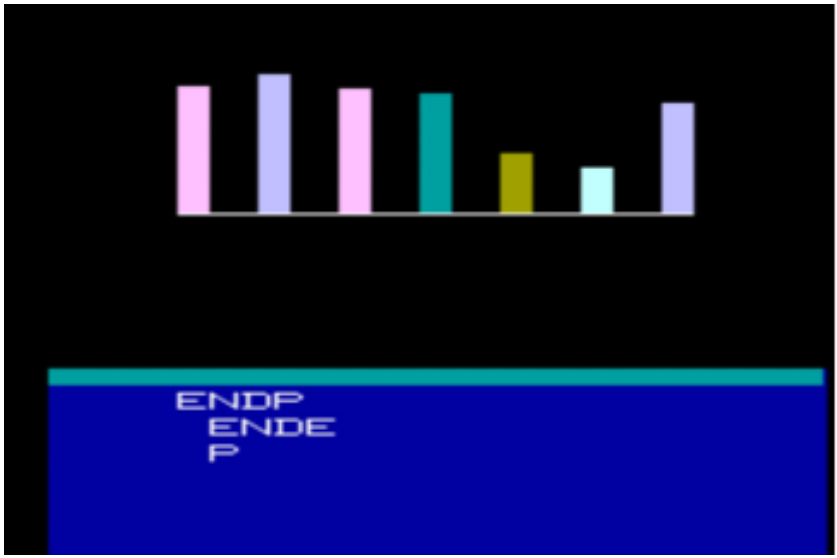

ENDP
ENDE
P

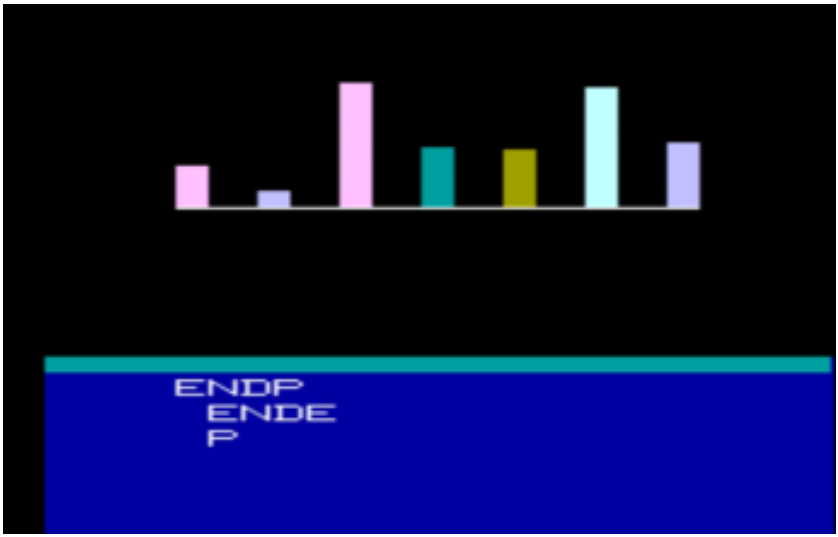

ENDP
ENDE
P

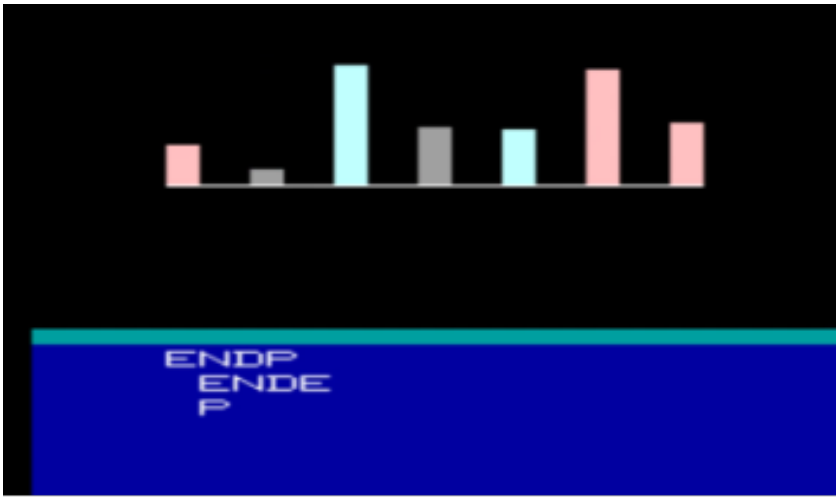

ENDP
ENDE
P

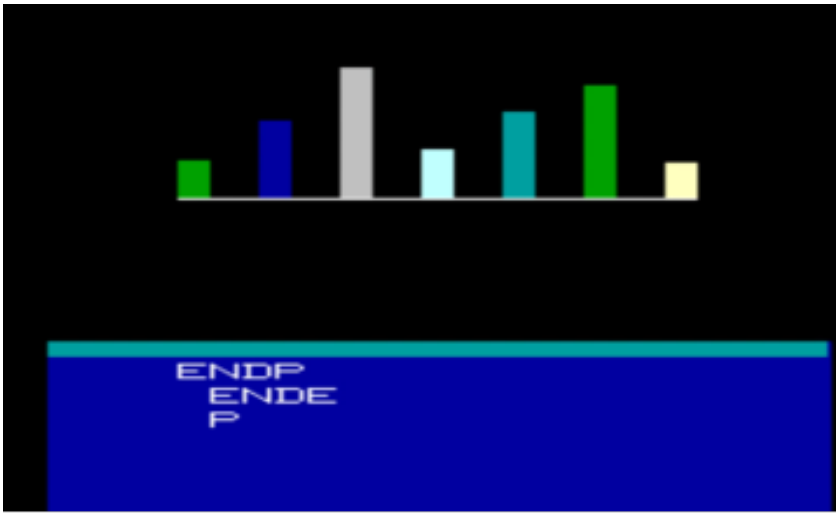

ENDP
ENDE
P


ENDP
ENDE
P


ENDP
ENDE
P


ENDP
ENDE
P


ENDP
ENDE
P

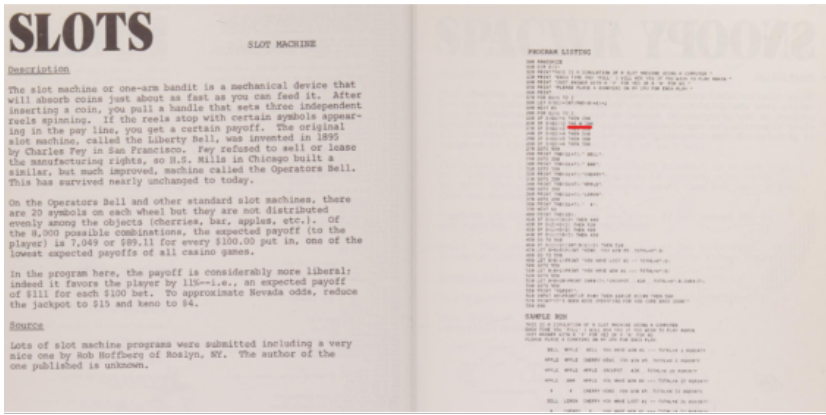

ENDP
ENDE
P

That should be enough information to get you started. I hope you have fun experimenting with the system, playing these games, and writing your own programs!

Slot Machine

A simplified port of the David Ahl Basic Computer Games SLOTS game.

```
PROG SLOT
CASH=100
BARH 20 20 20
WHILE 1
DO CASH
"BET $1?"
IF 0 GE BET=KEYB
THEN EXIT
ENDIF
BARC RAND 6 6 6
BARC RAND 6 6 6
BARC RAND 6 6 6
BARC RAND 6 6 6
BARC RAND 6 6 6
BARC RAND 6 6 6
Z=RAND 6 6 6
BARC Z
I=+/ (Z(1) EQ Z)
IF I EQ 3
THEN
   "3 OF A KIND"
   CASH=CASH+5
ELSE
IF I EQ 2
THEN
   "2 OF A KIND"
   CASH=CASH+3
   ELSE
      J=(+/(Z(3) EQ Z))
      IF J EQ 2
      THEN
         "2 OF A KIND"
         CASH=CASH+3
      ENDIF
   ENDIF
ENDIF
CASH=CASH-1
ENDWHILE
ENDP
```



SLOTS

SLOT MACHINE

Description

The slot machine or one-arm bandit is a mechanical device that will absorb coins just about as fast as you can feed it. After inserting a coin, you pull a handle that sets three independent rvels spinning. If the reels stop with certain symbols appearing in the pay line, you get a certain payoff. The original slot machine, called the Liberty Bell, was invented in 1895 by Charles Fey in San Francisco. Fey refused to sell or lease the manufacturing rights, so H.S. Mills in Chicago built a similar, but much improved, machine called the Operators Bell. This has survived nearly unchanged to today.

On the Operators Bell and other standard slot machines, there are 20 symbols on each wheel but they are not distributed evenly among the objects (cherries, bar, apples, etc.). Of the 8,000 possible combinations, the expected payoff (to the player) is 7.049 or $89.11 for every $100.00 put in, one of the lowest expected payoffs of all casino games.

In the program here, the payoff is considerably more liberal; indeed it favors the player by 11%--i.e., an expected payoff of $111 for each $100 bet. To approximate Nevada odds, reduce the jackpot to $15 and keno to $4.

Source

Lots of slot machine programs were submitted including a very nice one by Rob Hoffberg of Roslyn, NY. The author of the one published is unknown.

PROGRAM LISTING

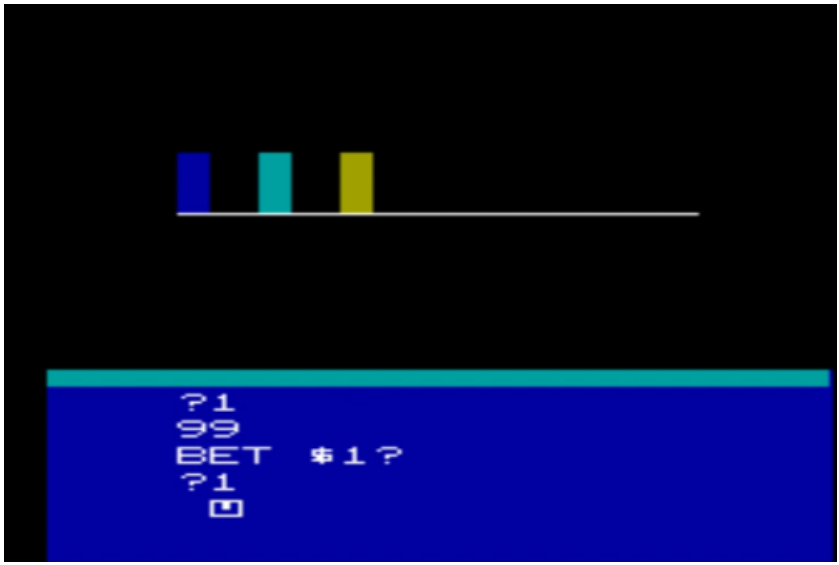
SAMPLE RUN

An error in the original listing.

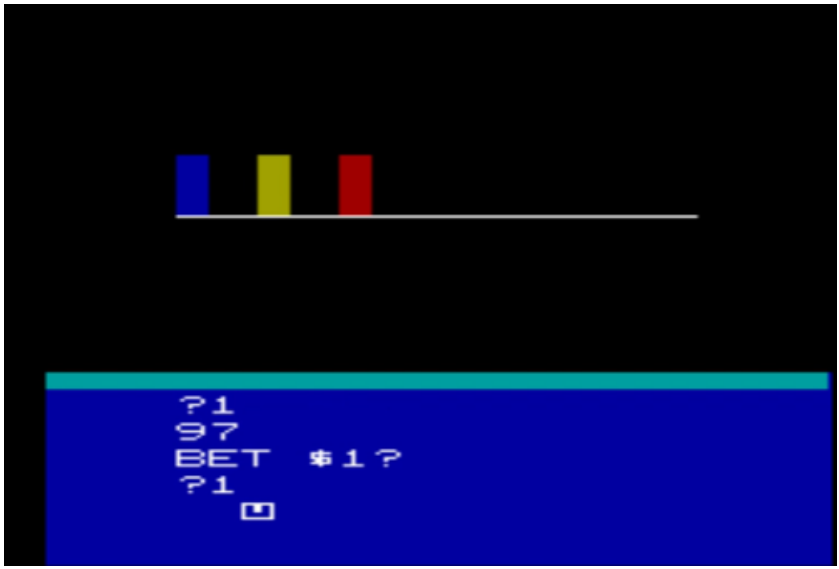
PROGRAM LISTING

```
100 RANDOMIZE
110 DIM D(3)
120 PRINT"THIS IS A SIMULATION OF A SLOT MACHINE USING A COMPUTER "
130 PRINT "EACH TIME YOU 'PULL' I WILL ASK YOU IF YOU WISH TO PLAY AGAIN."
140 PRINT "JUST ANSWER WITH A 'Y' FOR YES OR A 'N' FOR NO."
150 PRINT "PLEASE PLACE 4 QUARTERS ON MY CPU FOR EACH PLAY."
160 PRINT
170 FOR B1=1 TO 3
180 LET D(B1)=INT(RND(0)*6)+1
190 NEXT B1
200 FOR G1=1 TO 3
210 IF D(G1)=1 THEN 280
220 IF D(G1)=2 THE N 300
230 IF D(G1)=3 THEN 320
240 IF D(G1)=4 THEN 340
250 IF D(G1)=5 THEN 360
260 IF D(G1)=6 THEN 380
270 GOTO 580
280 PRINT TAB(G1*7);" BELL";
290 GOTO 390
300 PRINT TAB(G1*7);" BAR";
310 GOTO 390
320 PRINT TAB(G1*7);"CHERRY";
330 GOTO 390
340 PRINT TAB(G1*7);"APPLE";
350 GOTO 390
360 PRINT TAB(G1*7);"LEMON";
370 GOTO 390
```

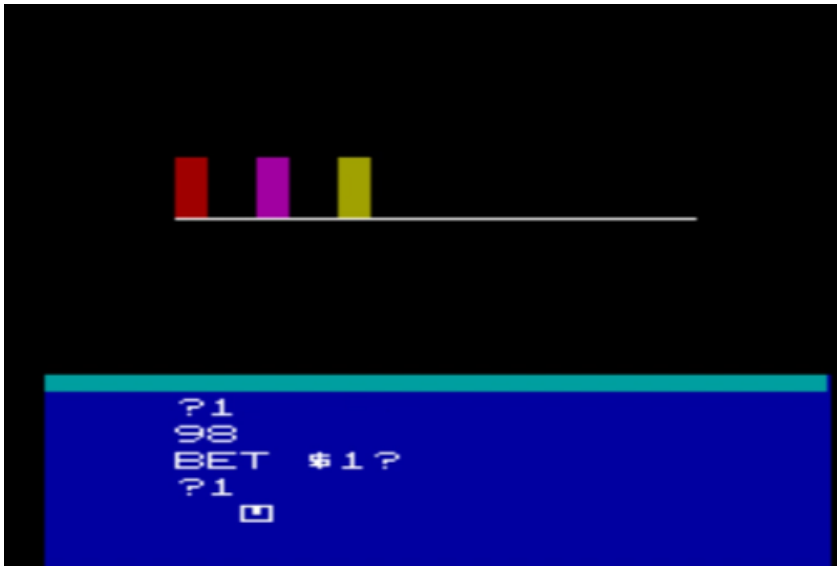
SLOT
100
BET $1?
?1


?1
99
BET $1?
?1

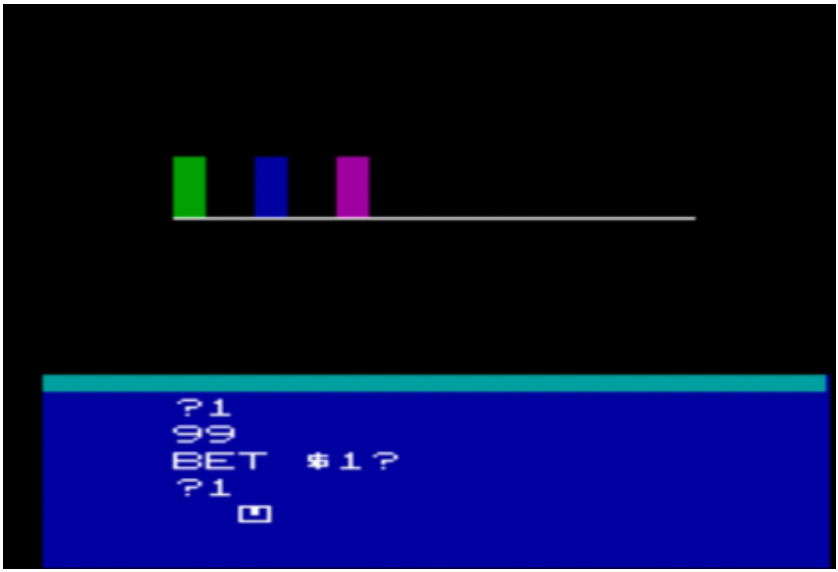
?1
98
BET ｻｲ1?
?1

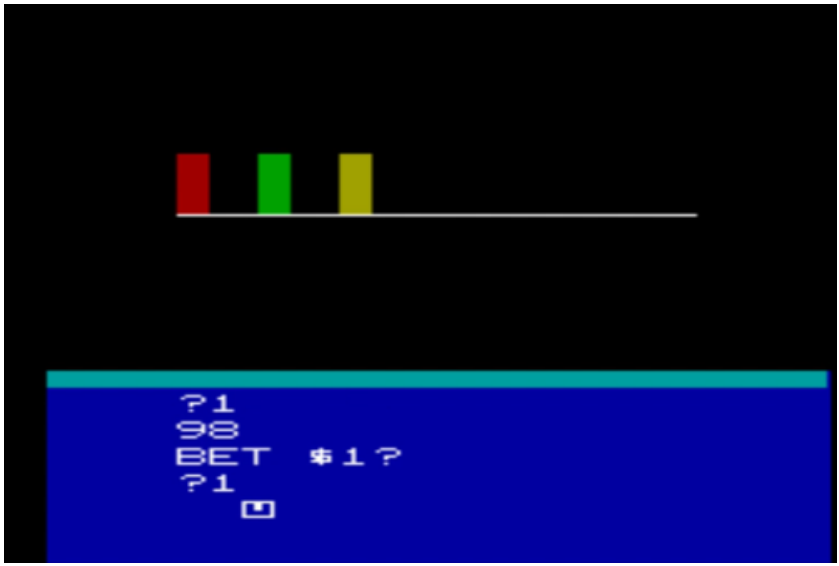

?1
97
BET $1?
?1

?1
2 OF A KIND
99
BET $1?

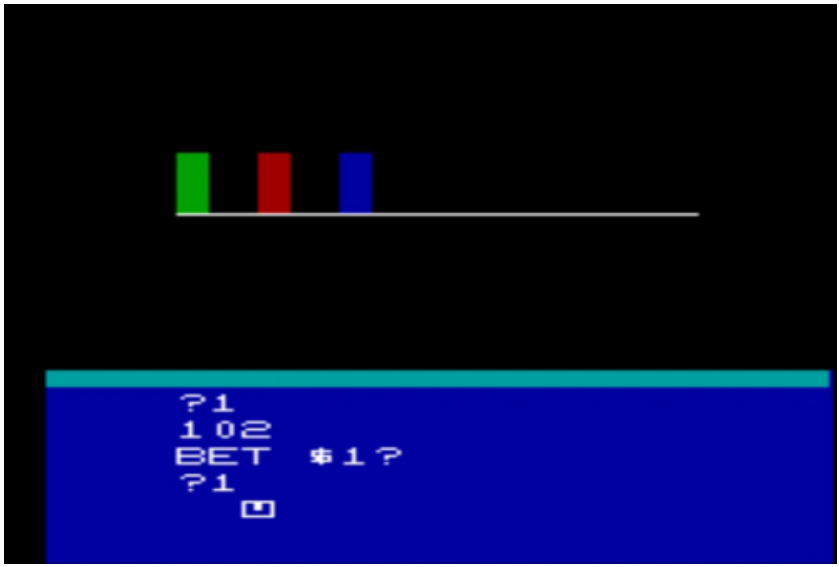

?1
98
BET $1?
?1

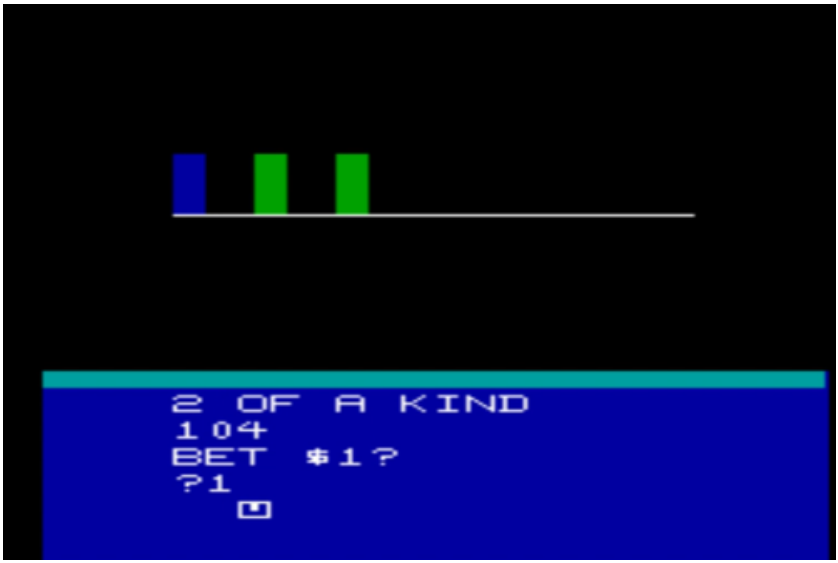
?1
2 OF A KIND
100
BET $1?


?1
99
BET $1?
?1

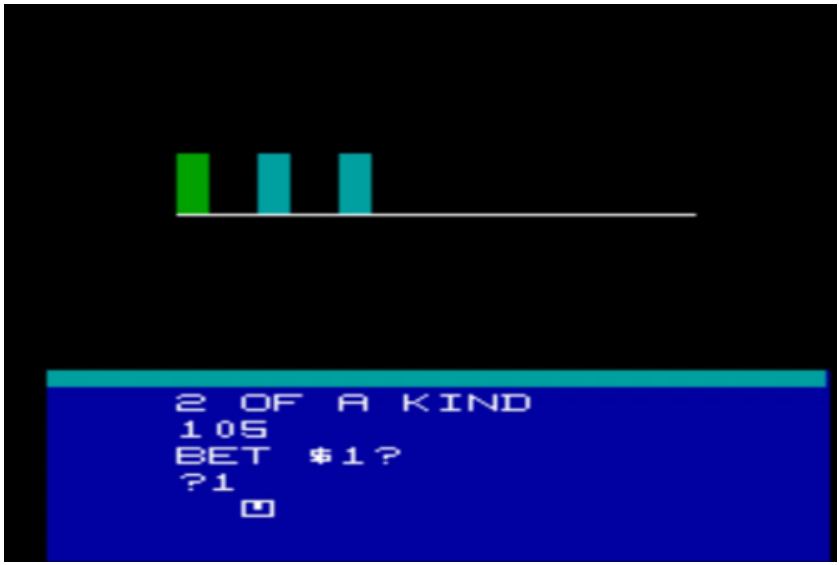

?1
98
BET $1?
?1

?1
3 OF A KIND
103
BET $1?

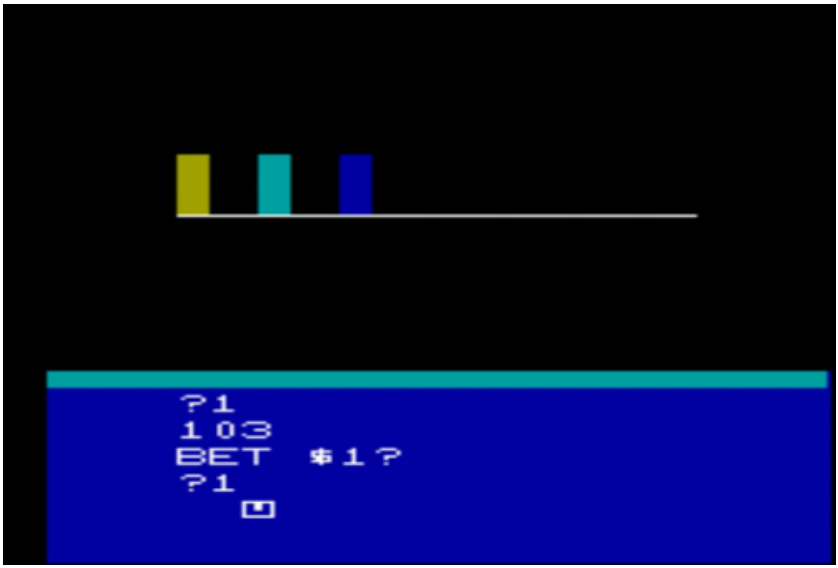

?1
102
BET $1?
?1


2 OF A KIND
104
BET $1?
?1

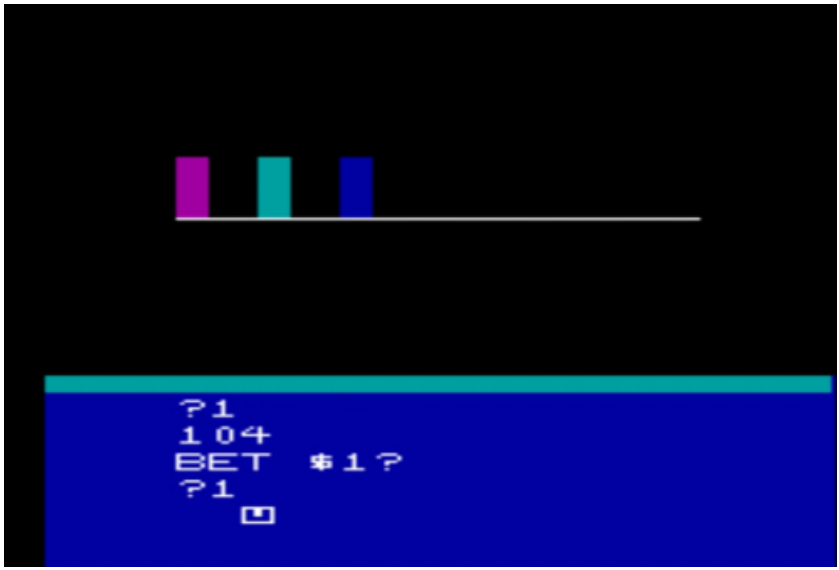
?1
103
BET $1?
?1


2 OF A KIND
10S
BET $1?
?1

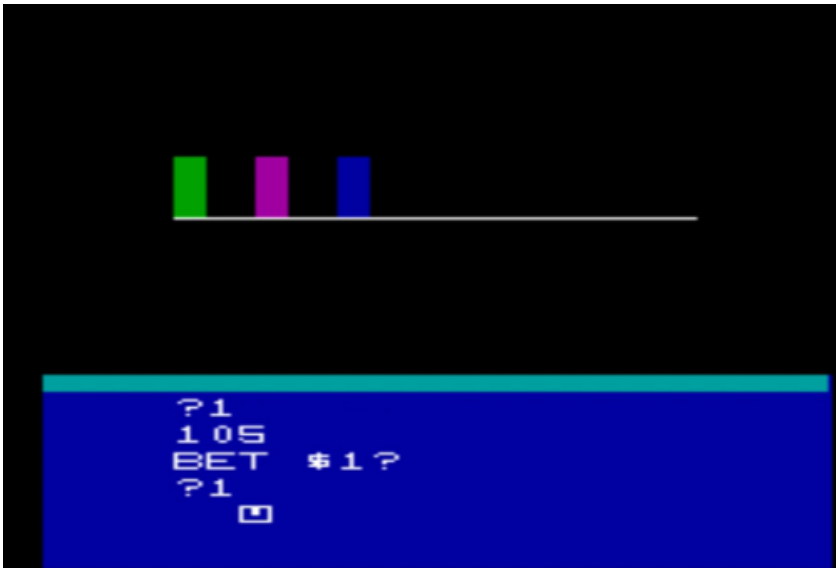
?1
104
BET $1?
?1


?1
103
BET $1?
?1

2 OF A KIND
105
BET $1?
?1

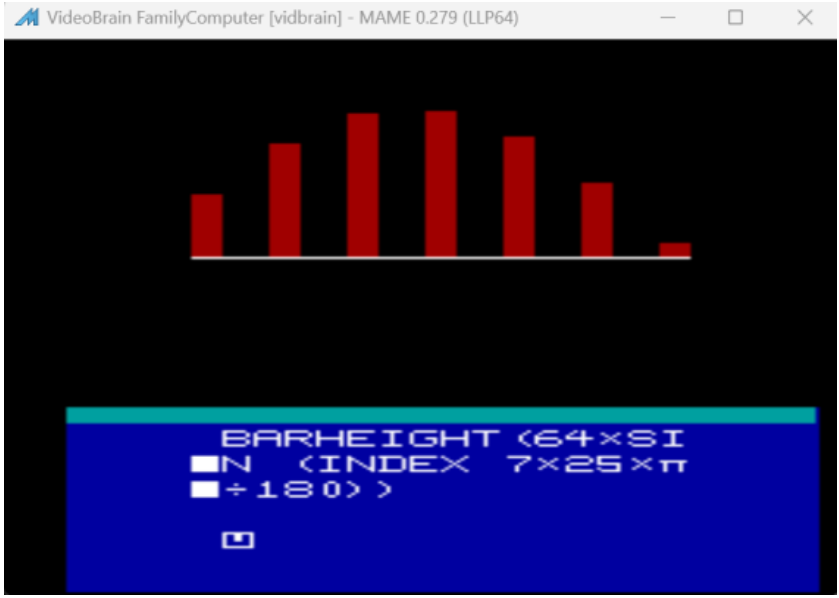

?1
104
BET $1?
?1

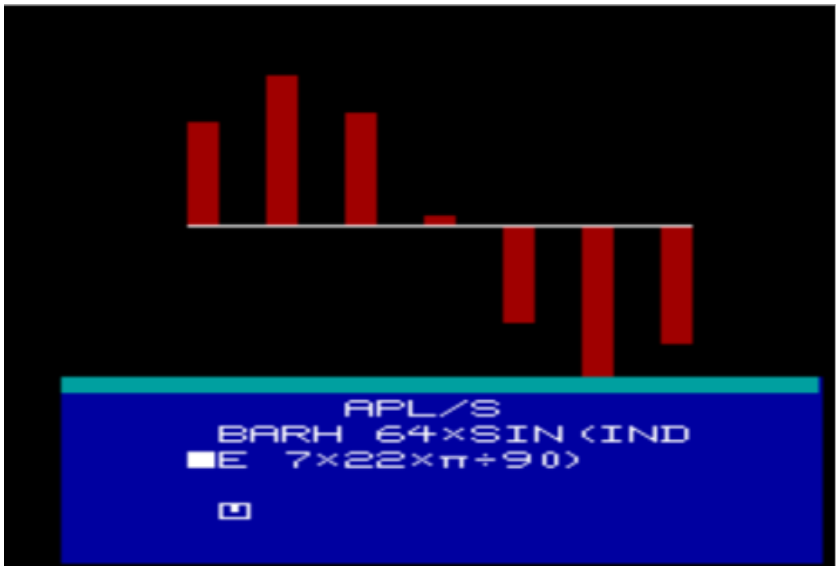
?1
2 OF A KIND
106
BET $1?

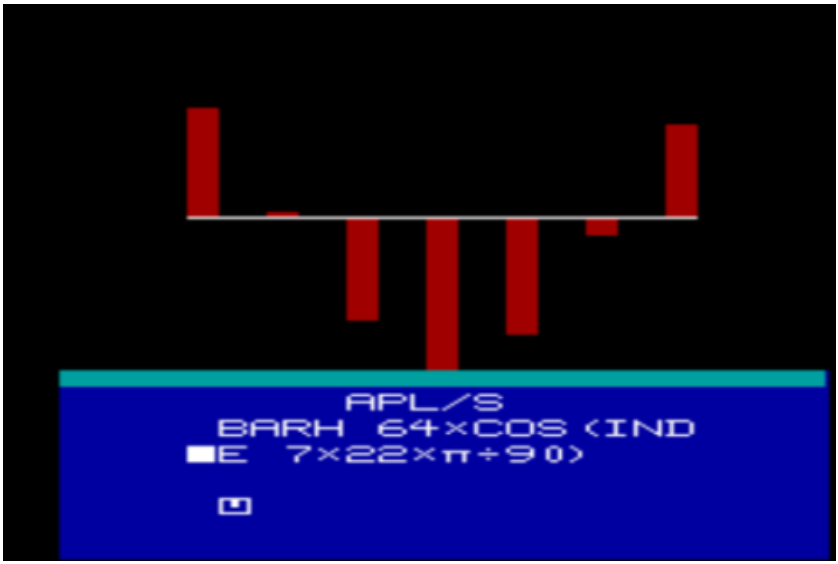

?1
105
BET $1?
?1

2 OF A KIND
107
BET $1?
?

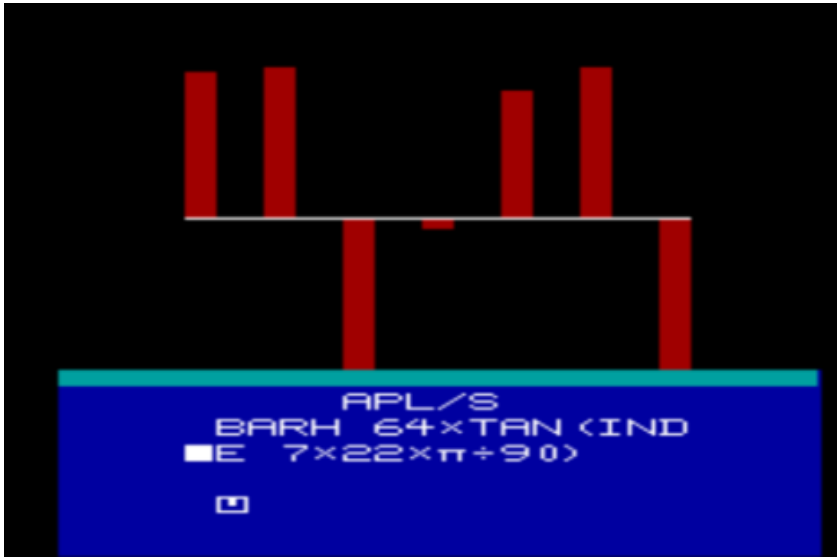
Sine Barchart One-Liner



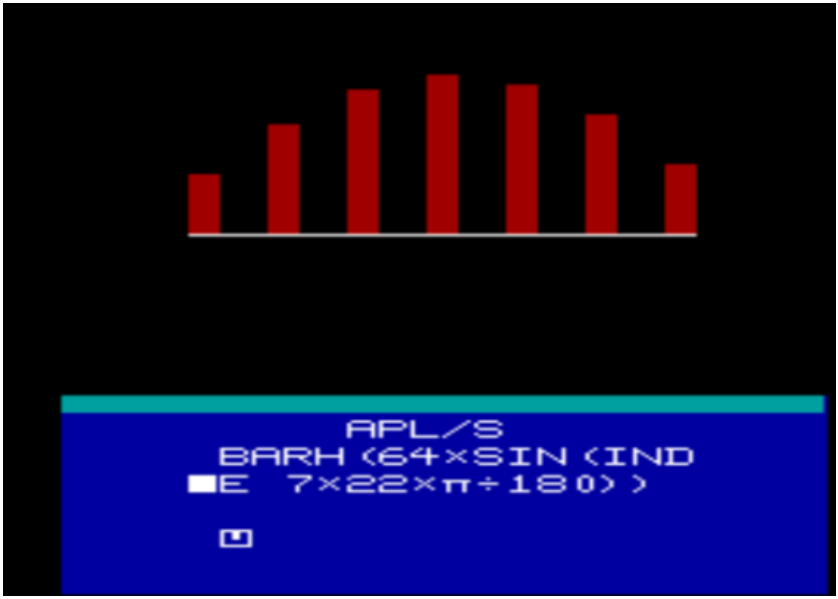

APL/S
BARH 64×SIN⍳IND
E 7×22×π÷90)


APL/S
BARH 64×COS (IND
E 7×22×π÷90)

Cosine bar charted similarly.



Tangent graphed at the same points.



Change the sine chart's pi divisor from 90 to 180 to focus only on the positive value portion.

Let’s walk through it, step by step.

Sample the sine curve using 7 values between 0 and 180 degrees.

Now convert from degrees to radians by multiplying by pi and dividing by 180.

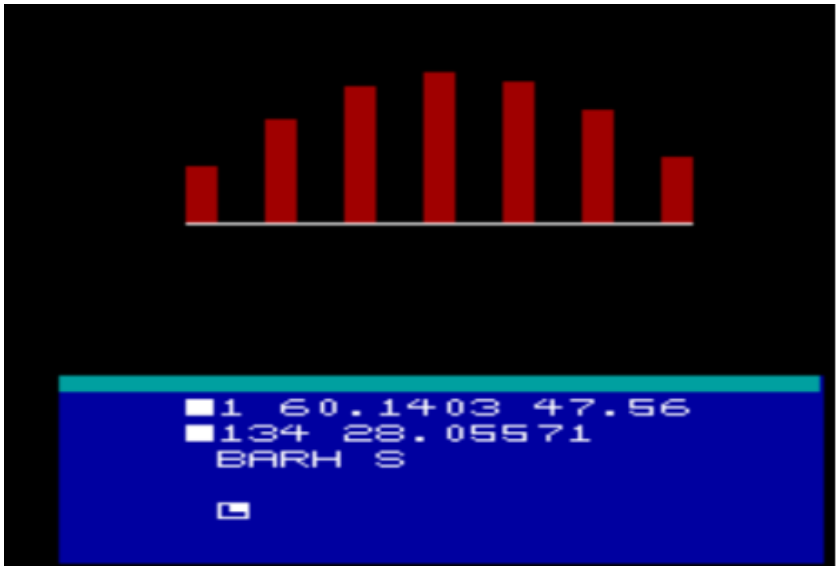
.303834 2.68780
8
S=SIN S

Take the sine of each of the 7 values in the array.

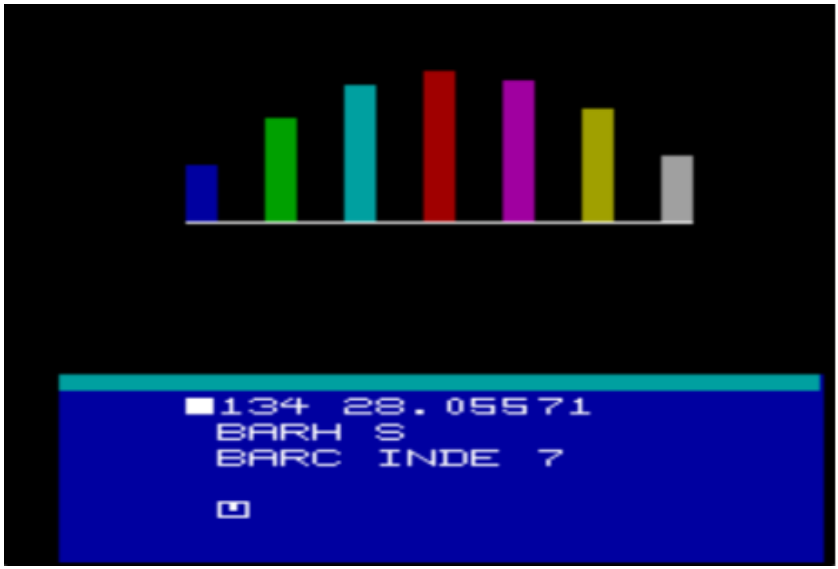
Scale up to 64 since the bar chart holds values between -64 and 64.

After scaling.

3 58.4669 63.96
1 60.1403 47.56
134 28.05571


1 60.1403 47.56
134 28.05571
BARH S

Now chart the scaled values. They do not need to be integers.


134 28.05571
BARH S
BARC INDE 7

Change colors with a simple BARC using INDEX 7.

Acey Ducey

A port of David Ahl's 1973 ACEYDU from "101 BASIC Computer Games"

```
PROG ACEY
BARC 3 4 3
```

```
PROG ACEY
BARC 3 4 3
N=100
Q=100
WHILE Q GT 0
DO
   “YOUR CASH”
   Q
   "YOUR CARDS"
   A=RAND  13 13 13
   WHILE A(1) GE A(3)
   DO
       A=RAND 13 13 13
   ENDW
   Z=A(1)
   CARD
   Z=A(3)
   CARD
   BARH 4.5x(A(1),0,A(3))
   "BET?"
   M=KEYB
   IF M EQ 99
   THEN EXIT
   ENDIF
   Z=A(2)
   CARD
   IF (Z GT A(1)) AND (Z LT A(3))
   THEN “YOU WIN”
      Q=Q+M
      BARC 3 2 3
   ELSE “YOU LOSE”
      Q=Q-M
       BARC 3 4 3
   ENDIF
   BARH 4.5xA
ENDW
ENDP
```

The Acey Ducey main program.

```
PROG CARD
IF (Z NE 1) AND (Z LT 11)
THEN Z
ELSE
  IF  Z EQ 1
   THEN “ACE”
   ELSE
       IF Z EQ 11
       THEN “JACK”
       ELSE
         IF Z EQ 12
         THEN “QUEEN”
         ELSE “KING”
         ENDI
      ENDI
   ENDI
ENDI
ENDP
```

The main program calls the CARD subprogram to generate cards.

ACEY
YOUR CASH
100
YOUR CARDS

A sample game.

4
5
BET?
?1

?1
KING
YOU LOSE

YOUR CASH
99
YOUR CARDS

ACE
KING
BET?
?50

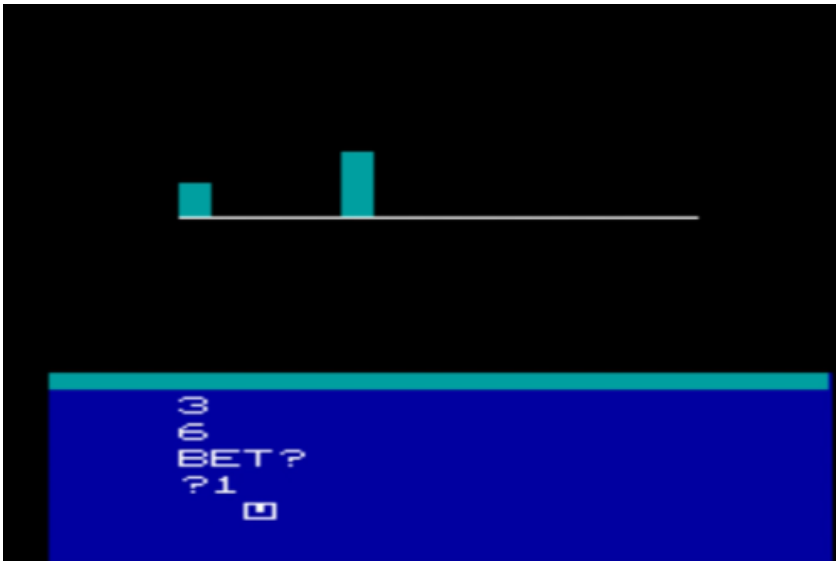
2
YOU WIN
YOUR CASH

149
YOUR CARDS
2
QUEEN

2
QUEEN
BET?
?100

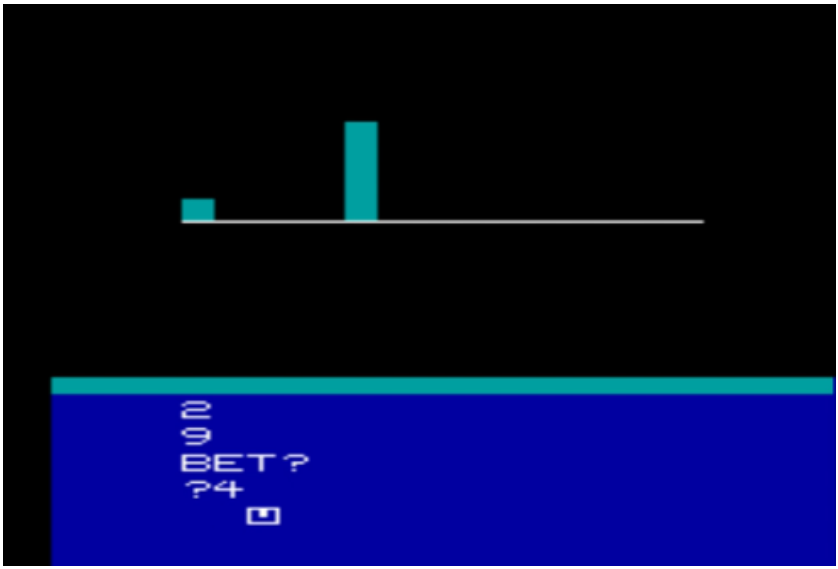
7
YOU WIN
YOUR CASH

YOUR CASH
249
YOUR CARDS


3
6
BET?
?1

?1
5
YOU WIN

YOUR CASH
25 0
YOUR CARDS

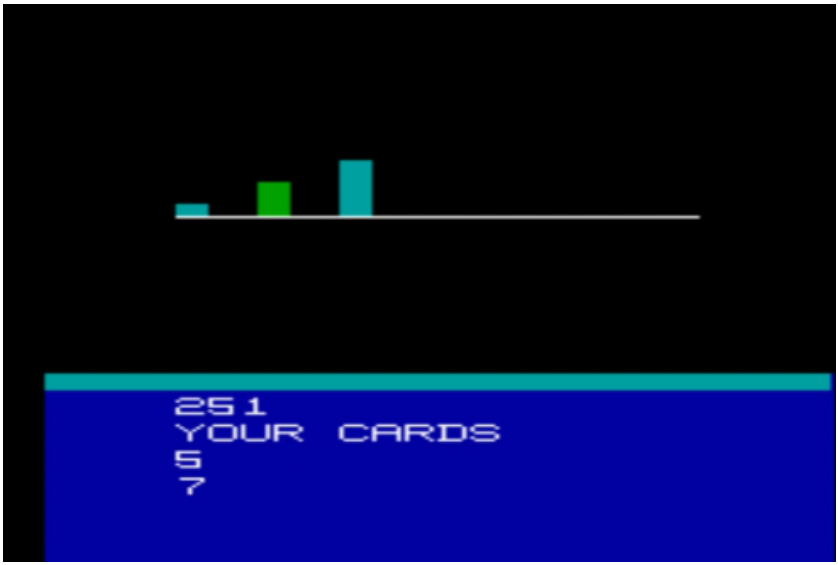

2
9
BET?
?4

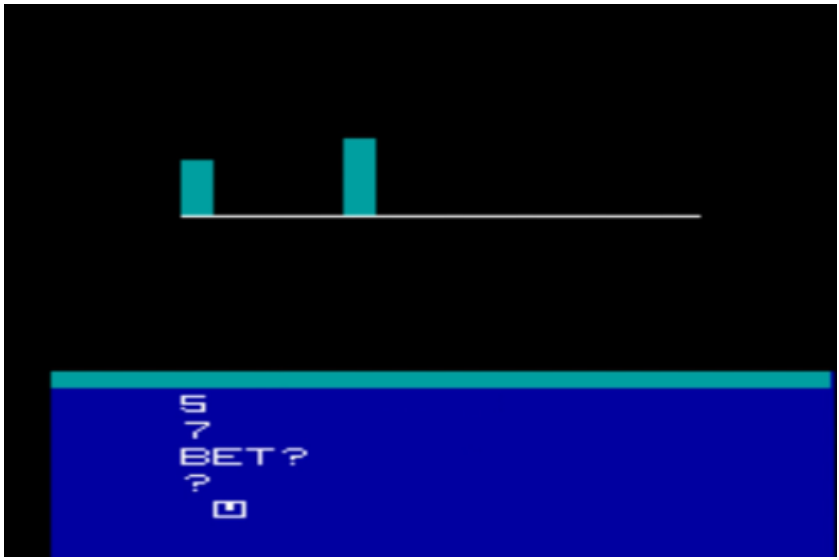
2
YOU LOSE
YOUR CASH

246
YOUR CARDS
ACE
5

ACE
5
BET?
?5

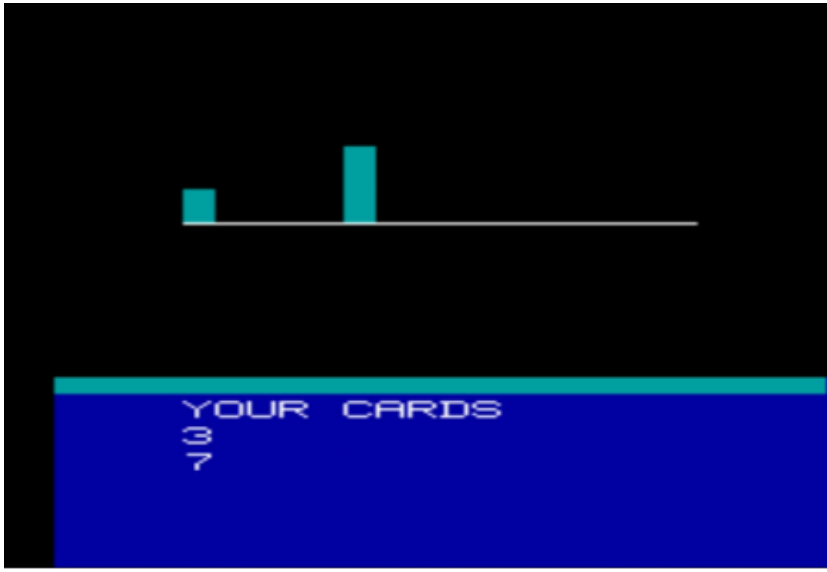
3
YOU WIN
YOUR CASH

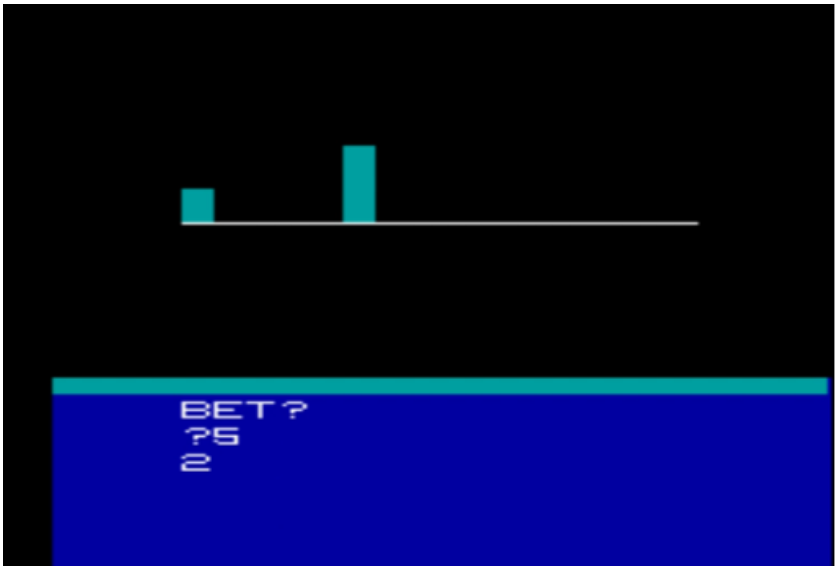

25 1
YOUR CARDS
5
7


5
7
BET?
?
0

ACEY
YOUR CASH
100

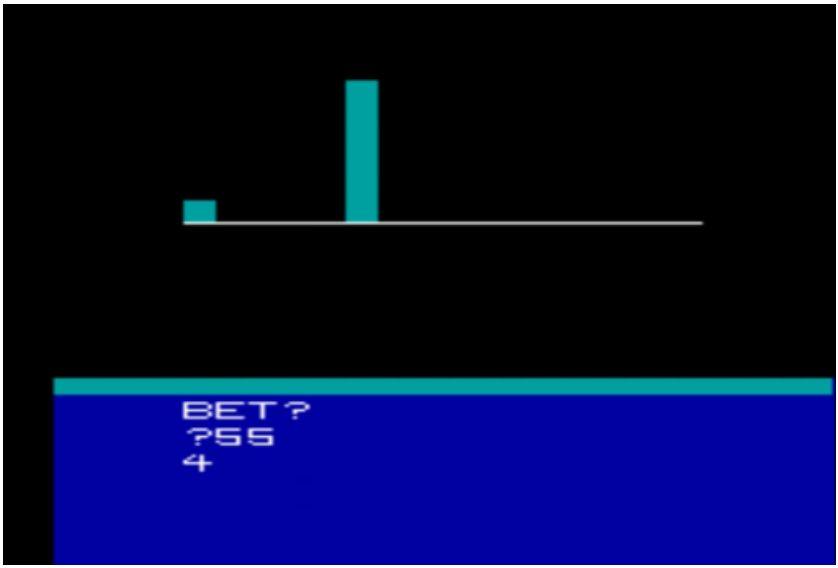
Another sample game.




BET?
?5
2

2
YOU LOSE
YOUR CASH
95

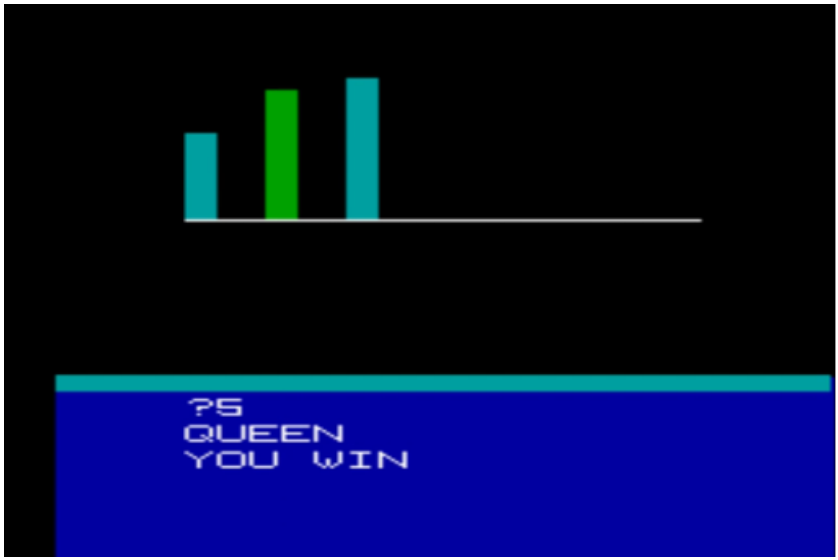
YOUR CARDS
2
KING


BET?
?55
4

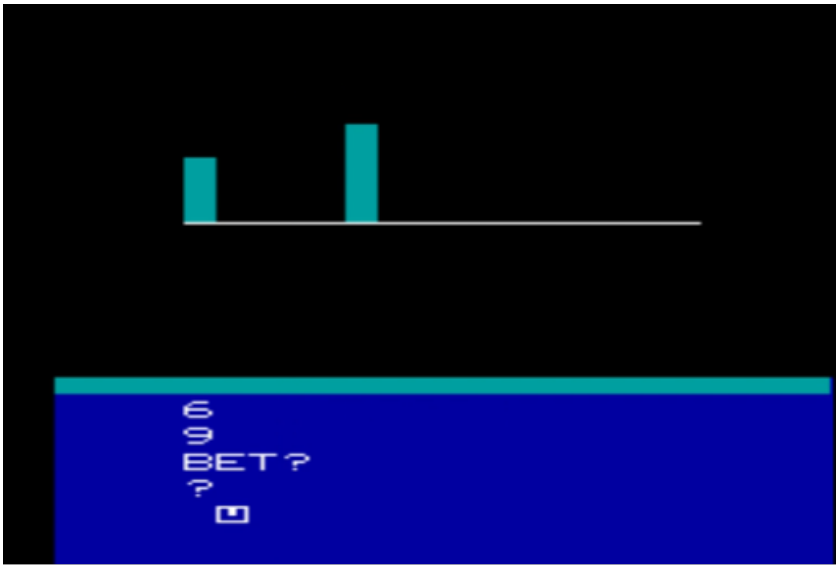
4
YOU WIN
YOUR CASH

150
YOUR CARDS
8

8
KING
BET?
?5


?5
QUEEN
YOU WIN

YOUR CASH
155
YOUR CARDS


6
9
BET?
?

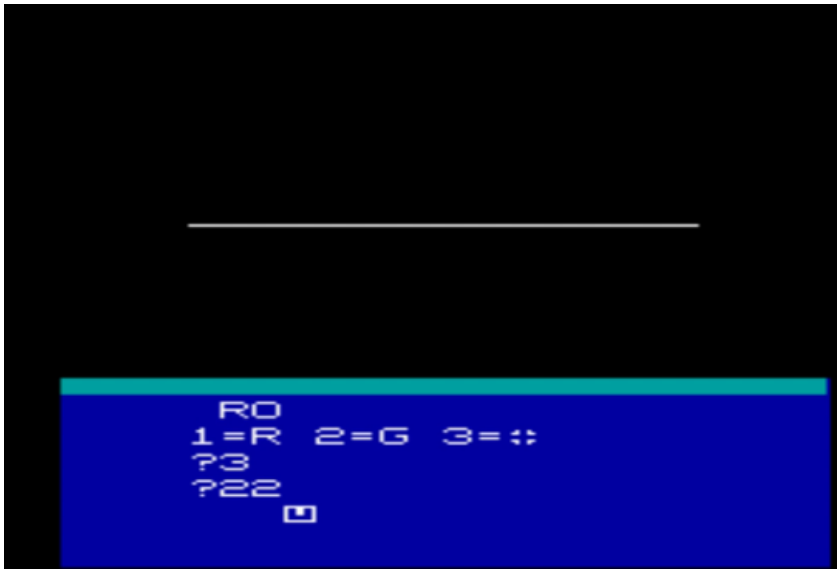
Roulette

First, enter the program. Then, before running, type:

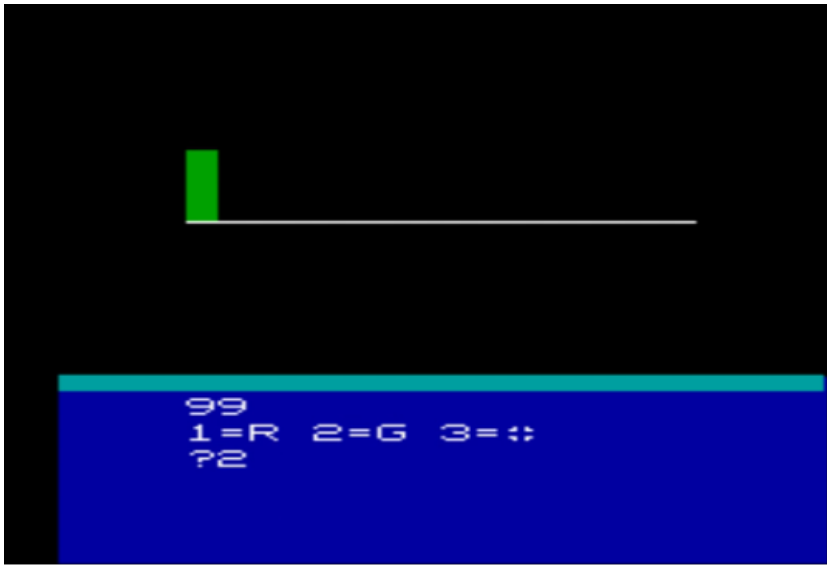
N = 0

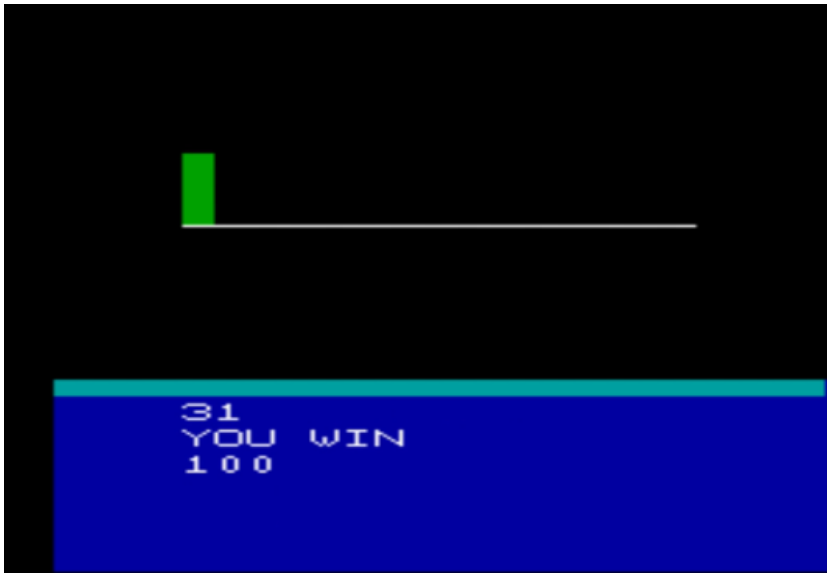
BARH 30

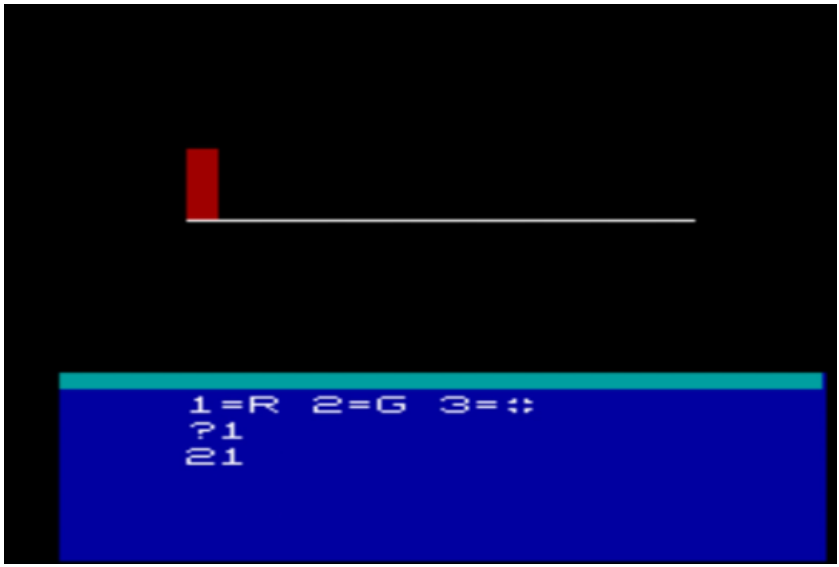
```
PROG RO
M=100
R=1 3 5 7 9 12 14 16 18 19 21 25 27 30 32 34 36
G=2 4 6 8 10 11 13 15 17 20 22 24 26 28 29 31 33 35
WHILE 1
DO
   M=M-1
   "1=R 2=G 3=#"
   P=KEYB
   IF P EQ 3
   THEN N=KEYB
      IF N EQ 99
      THEN EXIT
      ENDI
   ENDI
   S=RANDOM 37
   S=S-1
   S
   IF S EQ 0
   THEN BARC 8
      ELSE
      IF (+/(S EQ R))
      THEN BARC 4
      ELSE BARC 2
      ENDI
   ENDI
   IF (P EQ 3) AND (S EQ N)
   THEN "YOU WIN"
      M=M+35
      ELSE
      IF (P EQ 1) AND (+/(S EQ R))
      THEN "YOU WIN"
         M=M+2
         ELSE
         IF (P EQ 2) AND (+/(S EQ G))
         THEN "YOU WIN"
             M=M+2
             ELSE "YOU LOSE"
         ENDI
      ENDI
   ENDI
   M
ENDW
ENDP
```

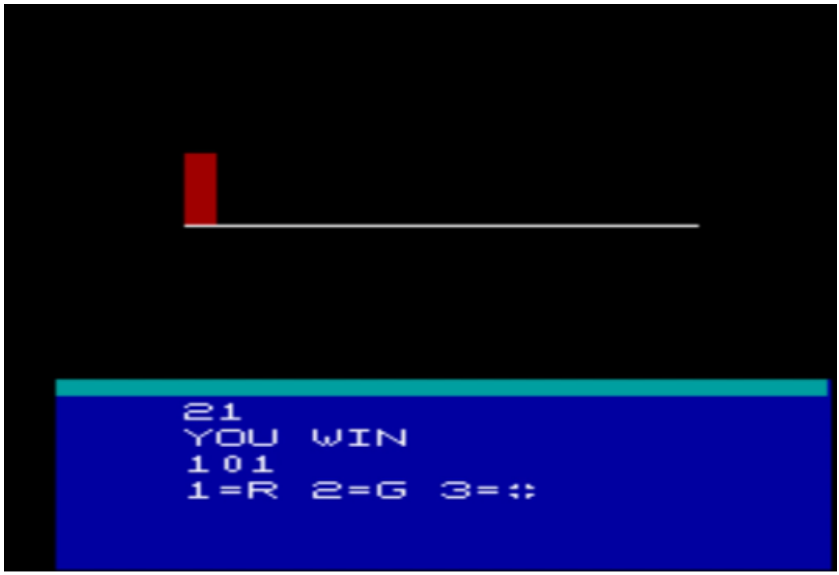


A sample game run.

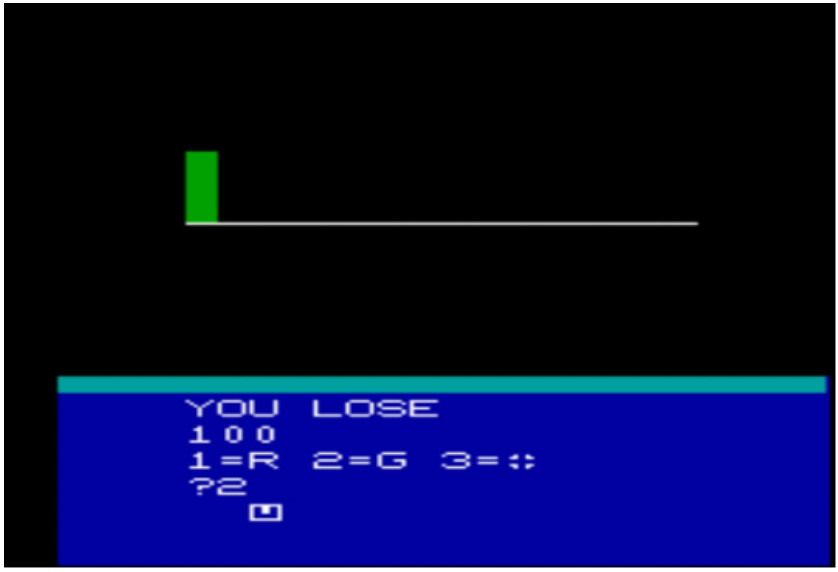

99
1=R 2=G 3=:
?2

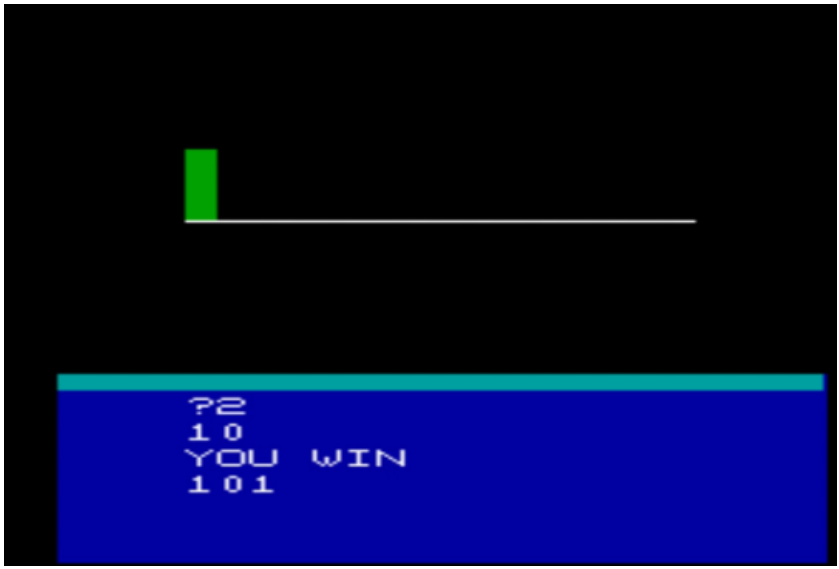

31
YOU WIN
100


1=R 2=G 3=:
?1
21


21
YOU WIN
101
1=R 2=G 3=:

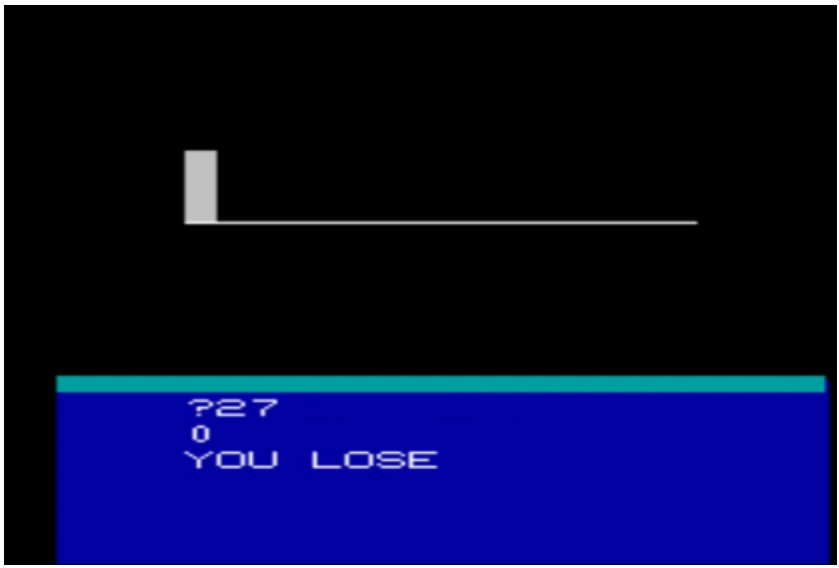
?1
20
YOU LOSE


YOU LOSE
100
1=R 2=G 3=:
?2



Exit the game by placing a wager on a roll of 99.

A big payout when the ball lands on the number the player wagered on.



The wheel rolls a 0. The house has the edge.

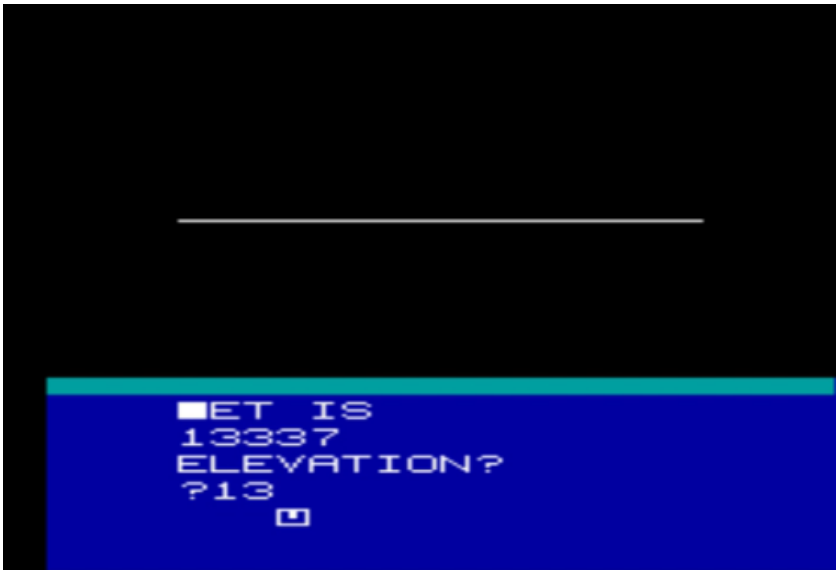
Gunner

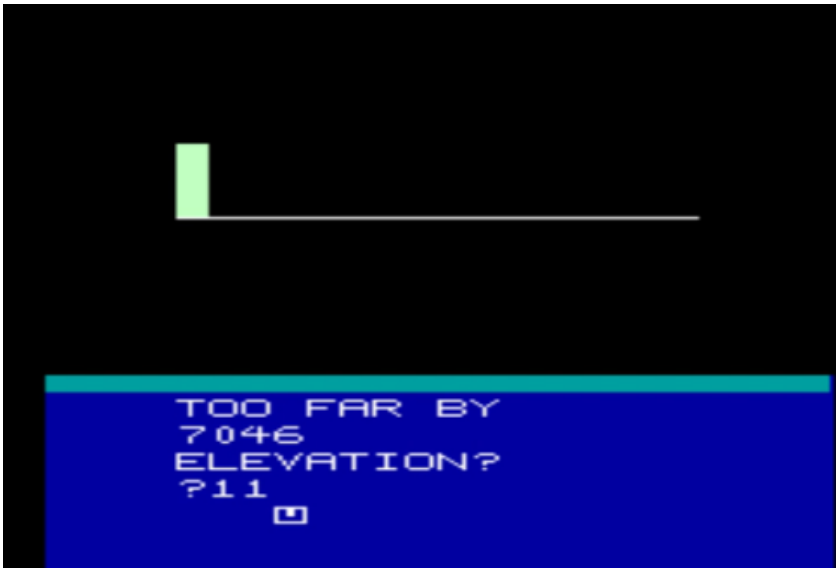
A port of David Ahl's 1973 GUNNER from "101 BASIC Computer Games"

```
PROG GUN
```

```
PROG GUN
Z=S=S1=0
BARC 10
T=43000-RAND 30000
R=46500
"DISTANCE TO TARGET IS"
T
WHILE 1
DO
   "ELEVATION?"
   B=KEYB
   B2=2×(B÷57.3)
   I=R×(SIN B2)
   X=T-I
   E=FLOOR X
   IF ABS E LT 100
   THEN "YOU WIN"
   EXIT
   ENDI
   IF E GT 100
   THEN "TOO SHORT BY"
      BAR -30
   ELSE
        "TOO FAR BY"
        BAR 30
   ENDI
   ABS E
   S=S+1
   IF S EQ 6
   THEN "YOU LOSE"
   EXIT
   ENDI
ENDW
ENDP
```

GUN
DISTANCE TO TARG
ET IS


■ET IS
13337
ELEVATION?
?13


TOO FAR BY
7046
ELEVATION?
?11

TOO FAR BY
4081
ELEVATION?
?7

TOO SHORT BY
2088
ELEVATION?
?8

TOO SHORT BY
520
ELEVATION?
?9

TOO FAR BY
1032
ELEVATION?
?8.5

TOO FAR BY
258
YOU LOSE

GUN
DISTANCE TO TARG
ET IS
26578

Another game.

TOO FAR BY
753
ELEVATION?
?17

TOO SHORT BY
577
ELEVATION?
?17.5

ELEVATION?
?17.5
YOU WIN

Collatz Bounce

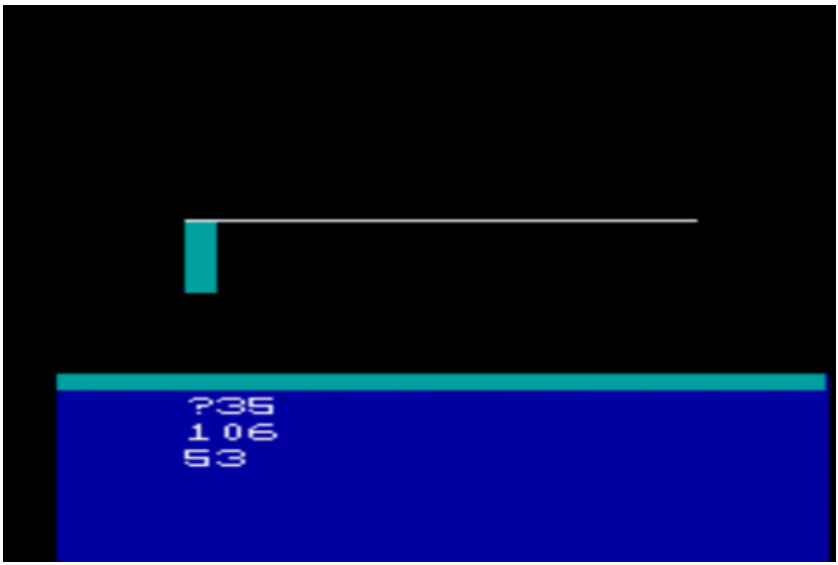
The Collatz conjecture is a long-standing open question in mathematics built around an extremely simple process. Starting from any positive integer, a sequence is generated by repeatedly applying one of two rules: if the current value is even, it is divided by two; if it is odd, it is multiplied by three and increased by one.

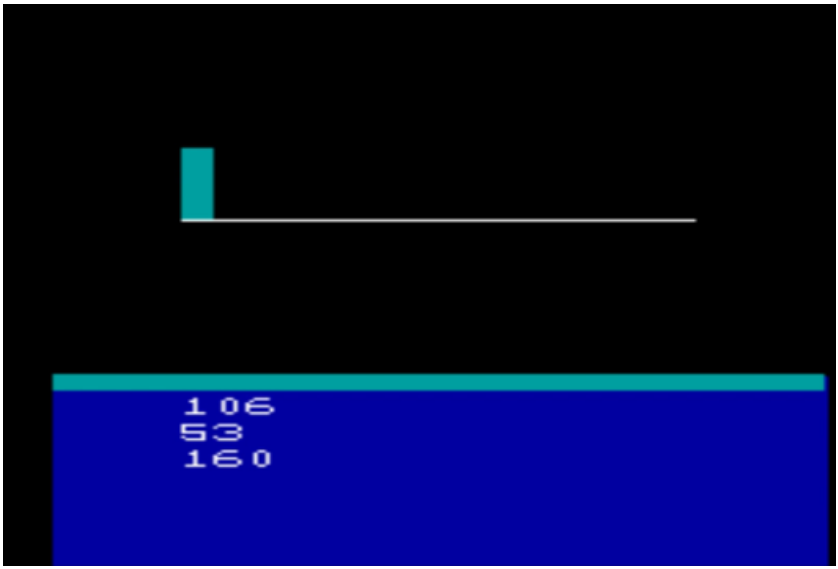
Despite the simplicity of these operations, no proof exists that the process always ends the same way. The conjecture asserts that, regardless of the starting number, the sequence will eventually reach the value 1, after which it falls into a repeating cycle. This claim has been verified for vast ranges of integers, but a general proof, or a counterexample, remains unknown.

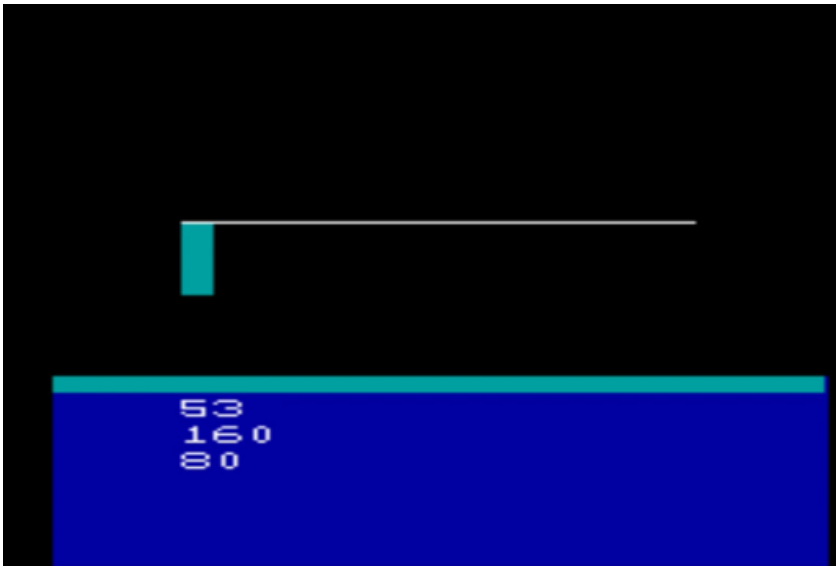
```
PROG COLLATZ
BARC 3
SCORE=0
"ENTER YOUR NUMBER TO COLLATZ BOUNCE"
NUM=I=KEYB
TOP=I
WHILE I NE 1
DO
   IF I MOD 2 EQ 0
   THEN I=I÷2
      BARH -30
   ELSE I=1+3×I
      BARH 30
   ENDI
   SCORE=SCORE+1
   I
   IF I GT TOP
   THEN TOP=I
   ENDI
ENDW
BARH 0
"NUM TOP SCORE"
NUM,TOP,SCORE
ENDP
```

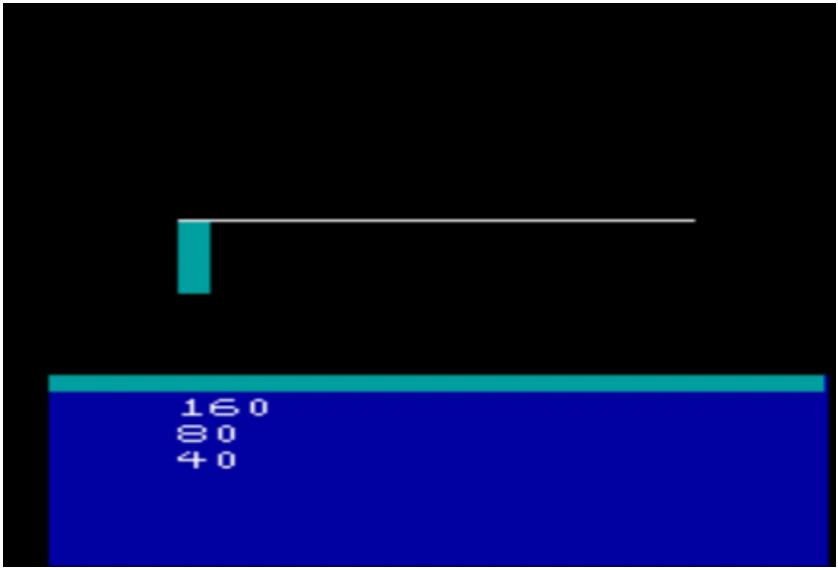
C
ENTER YOUR NUMBE
R TO COLLATZ BO
UNCE

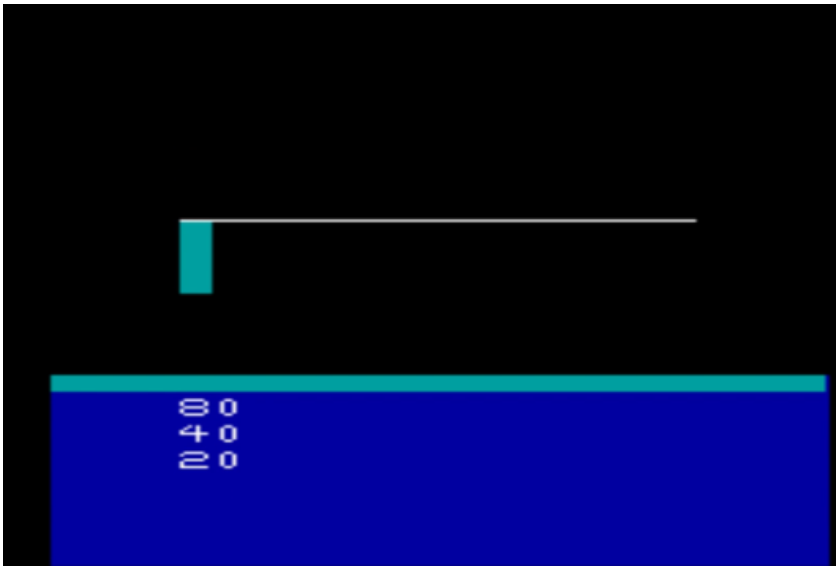
UNCE
?35
1 06

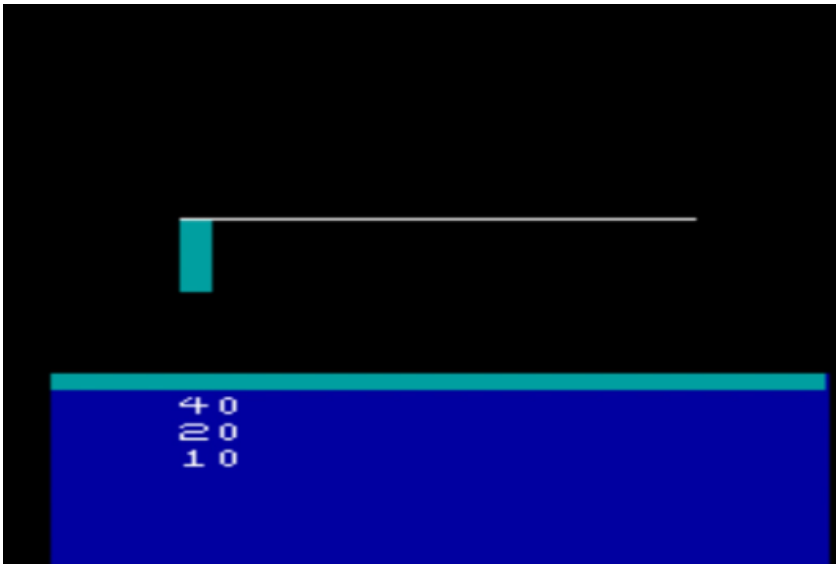

?35
1 06
53

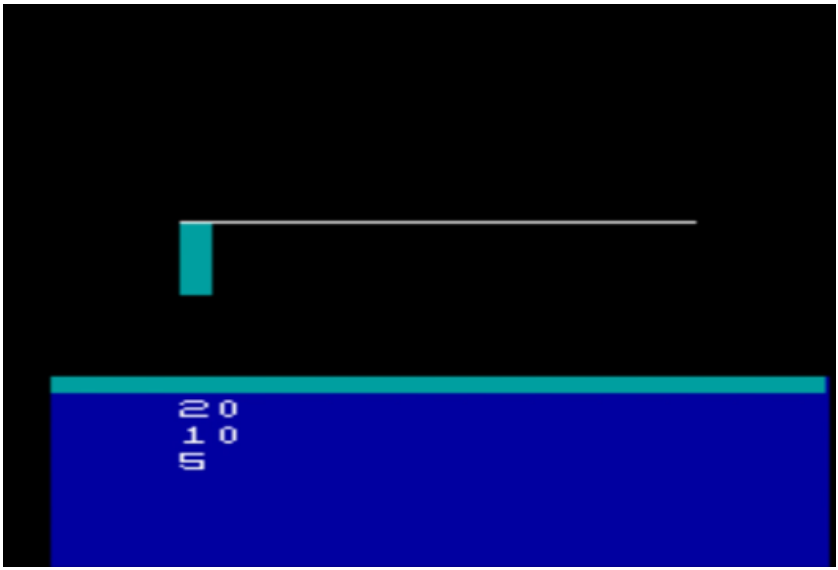

106
53
160

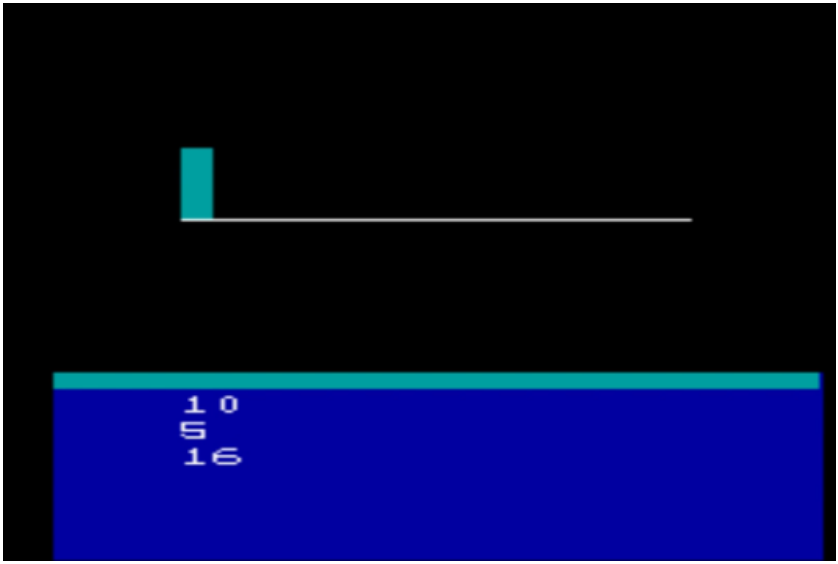

53
160
80

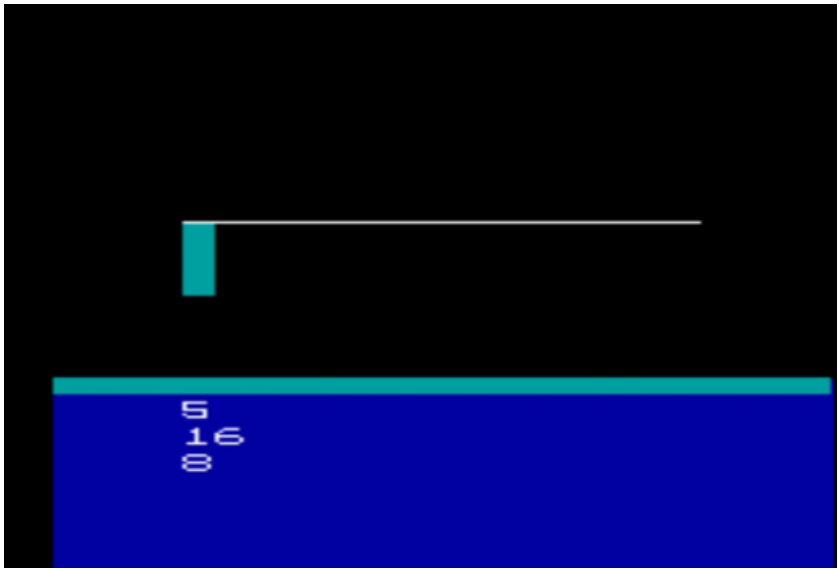

160
80
40

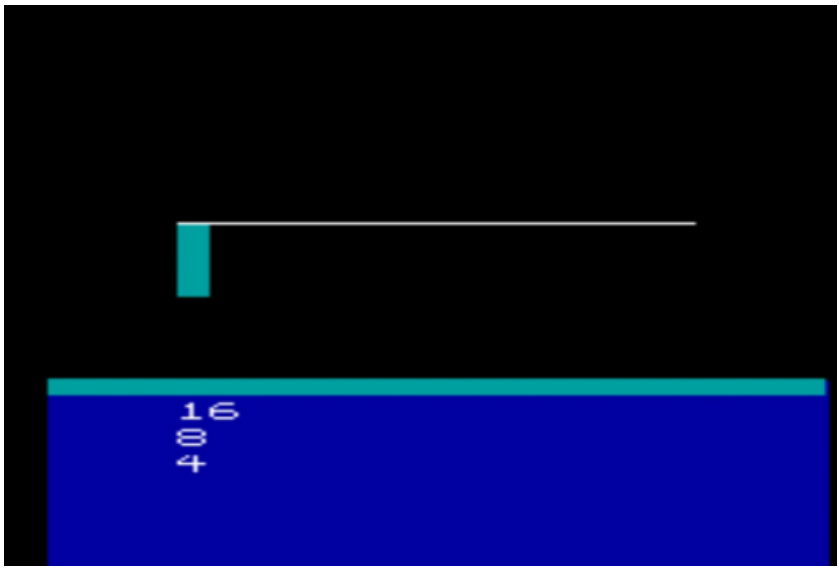

80
40
20

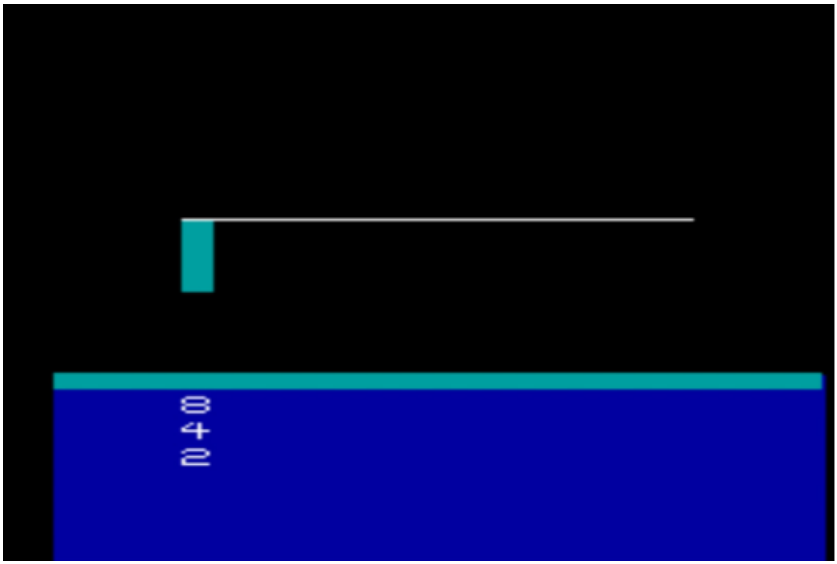

40
20
10

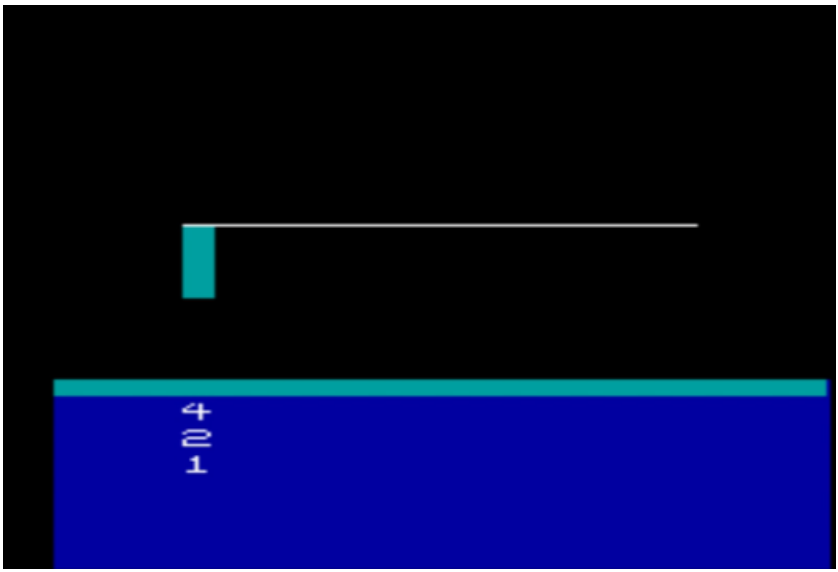

20
10
5


10
5
16


5
16
8

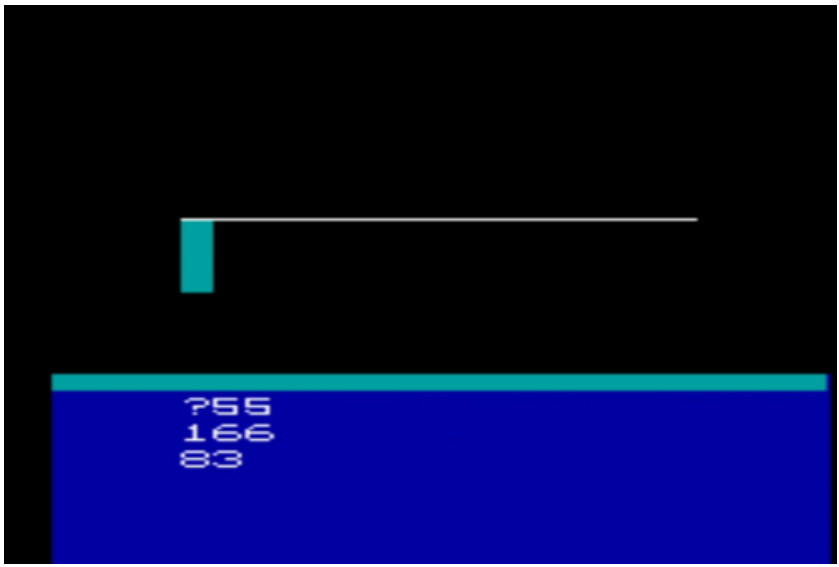

16
8
4

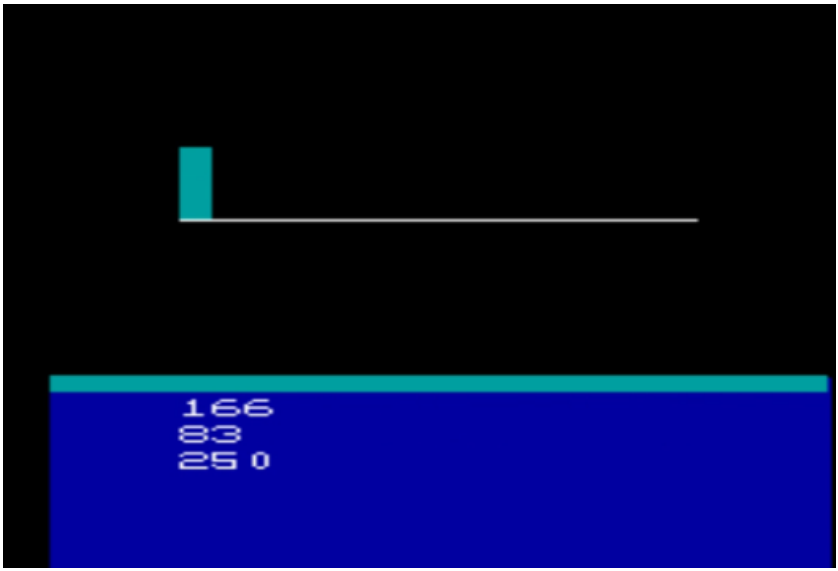

8
4
2

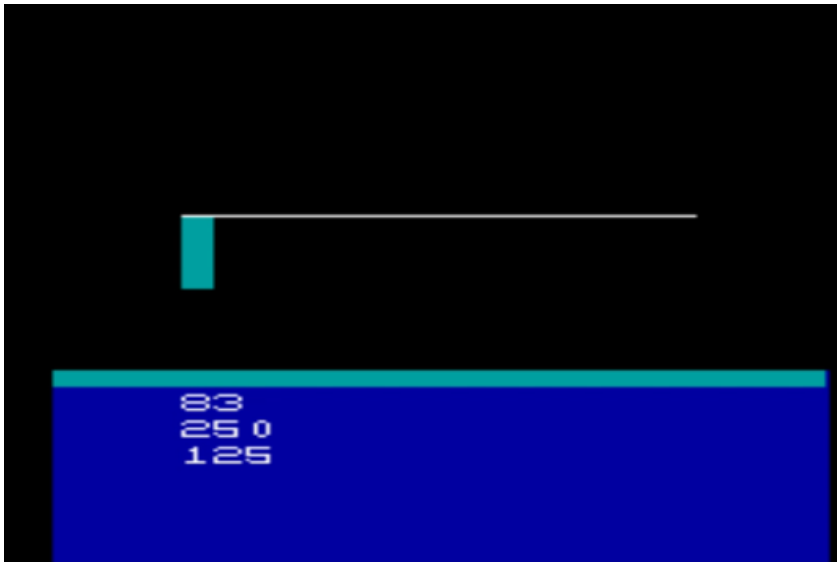

4
2
1

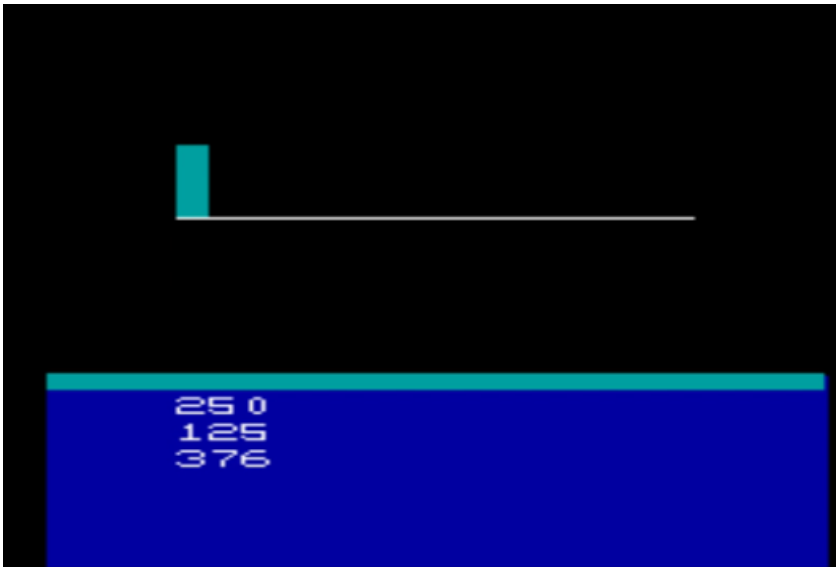
Another run, starting at 55.

UNCE
?55
166

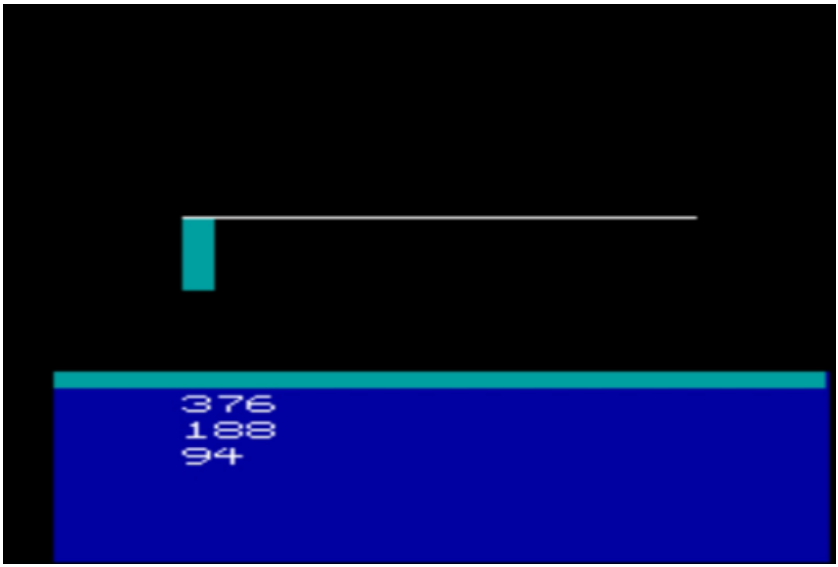

?55
166
83

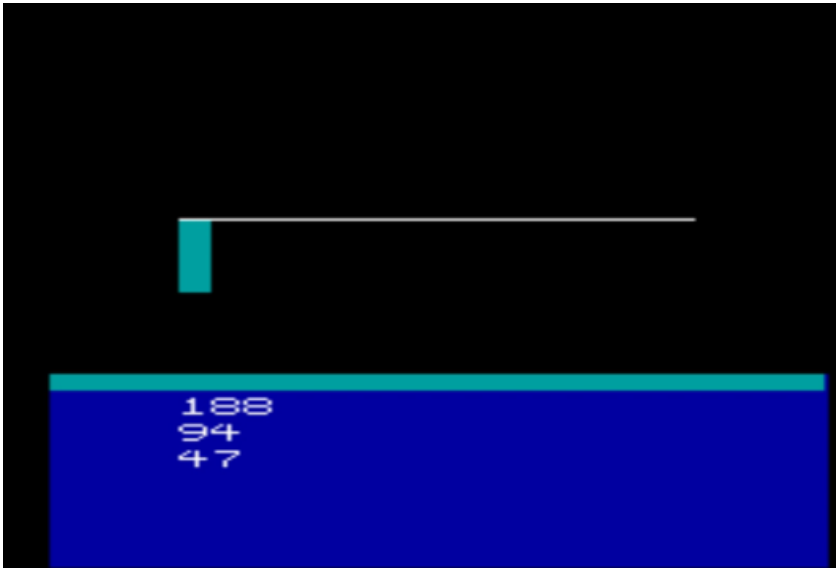

166
83
250

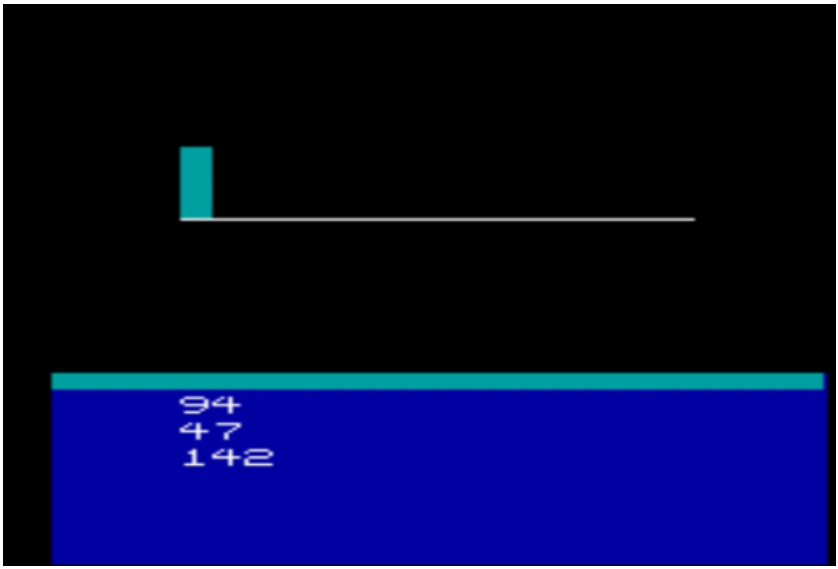

83
25 0
125

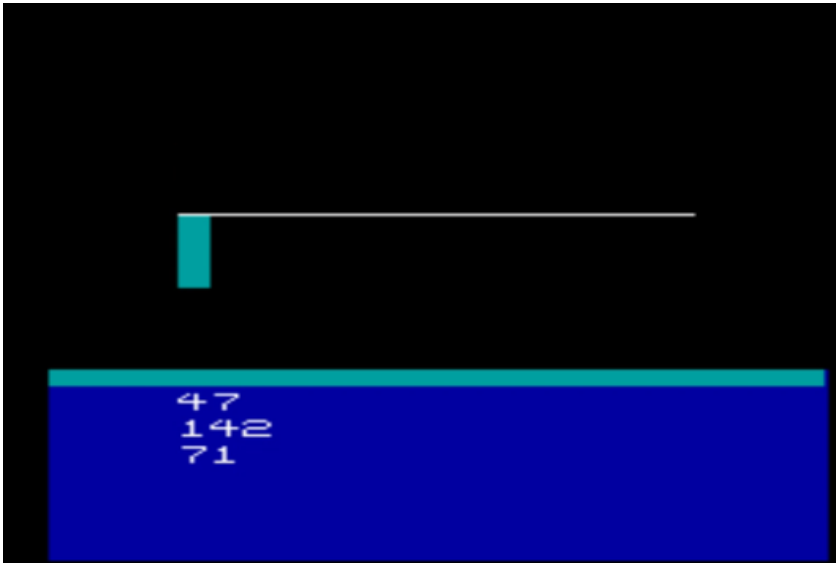

250
125
376

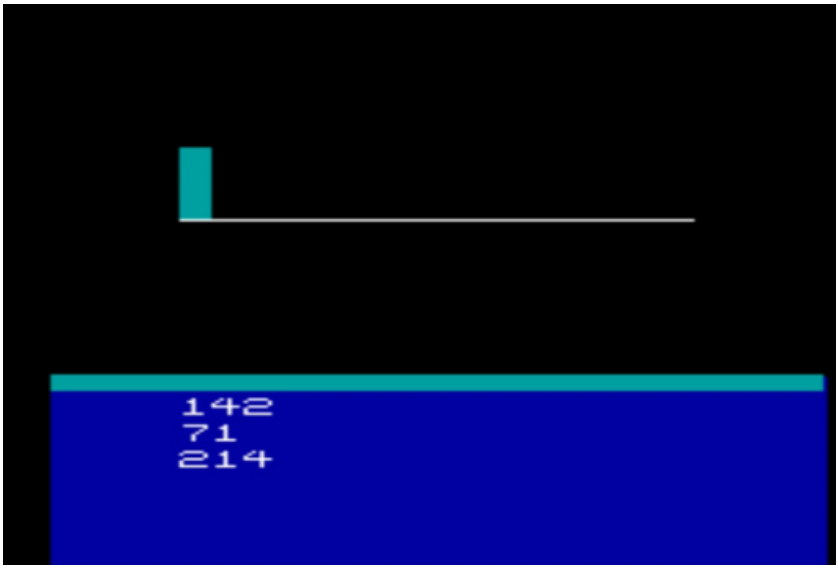
125
376
188

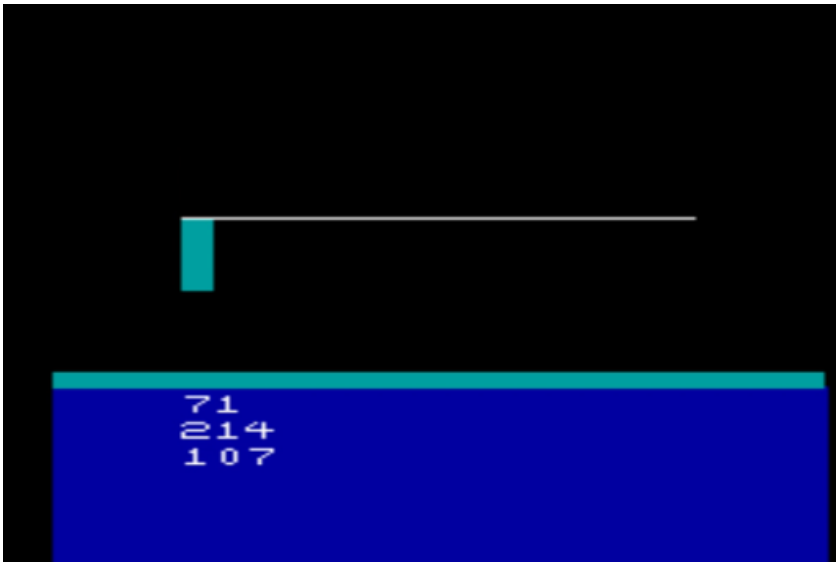

376
188
94

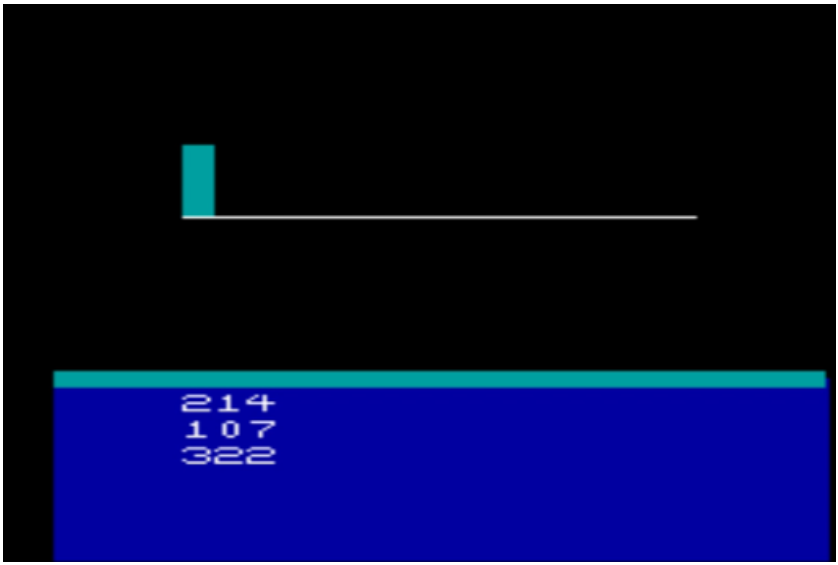

188
94
47

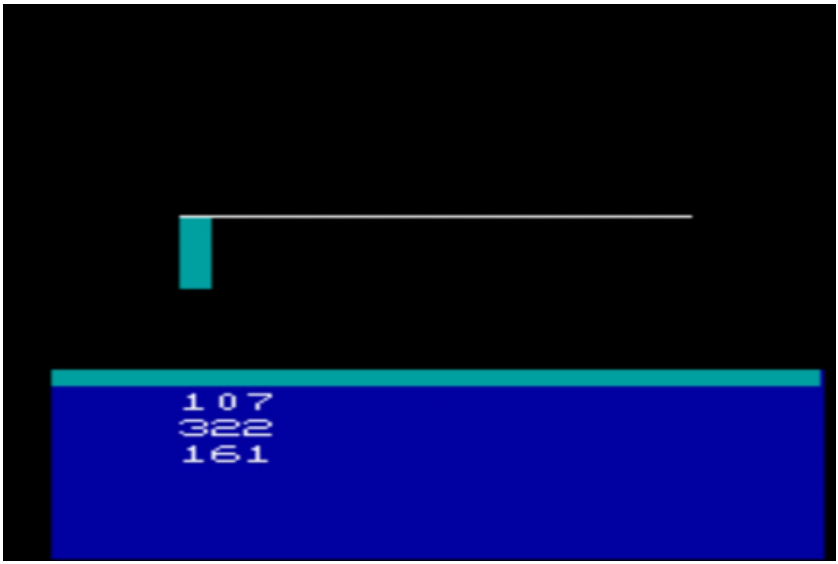

94
47
142

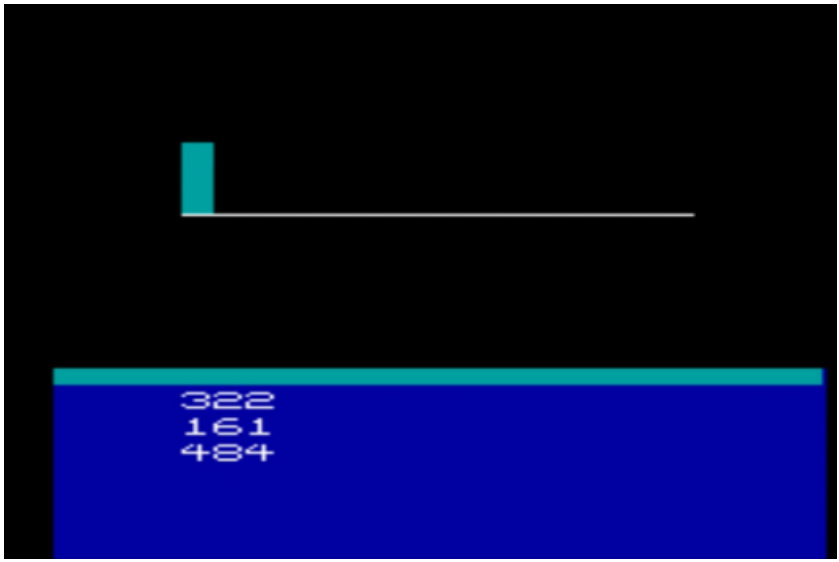

47
142
71

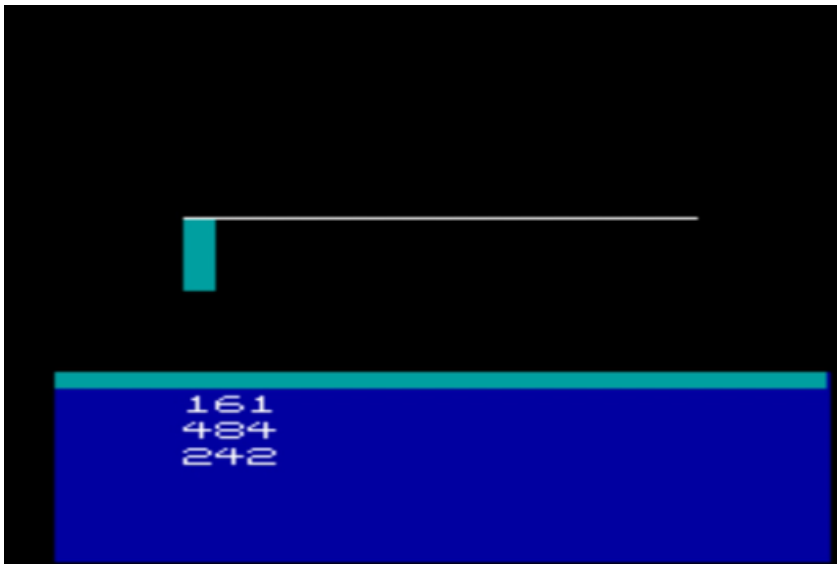

142
71
214


71
214
107


214
107
322


107
322
161


322
161
484


161
484
242

1
NUM TOP SCORE
55 9232 112

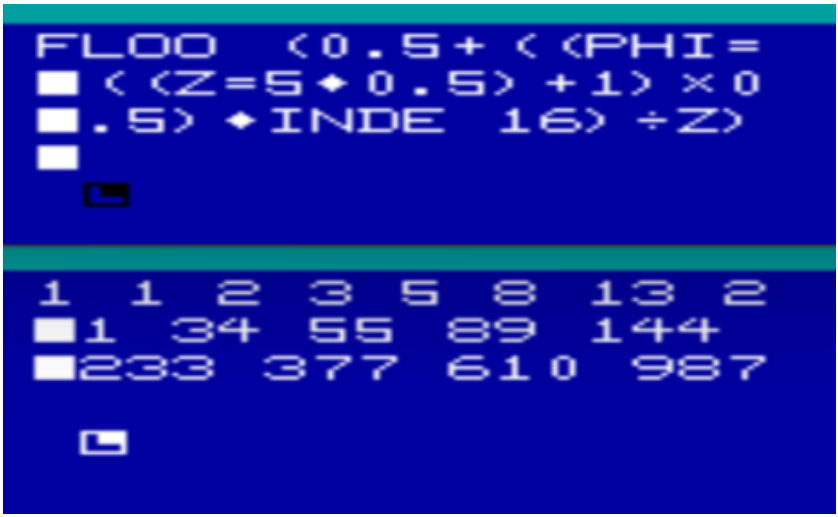
A maximum value of 9232 is reached, before settling back to 1 after over a hundred bounces.

Phi and Fibonacci One-Liners

The value of phi, 1.618034, is easy to calculate. It can then be used to generate the Fibonacci sequence.

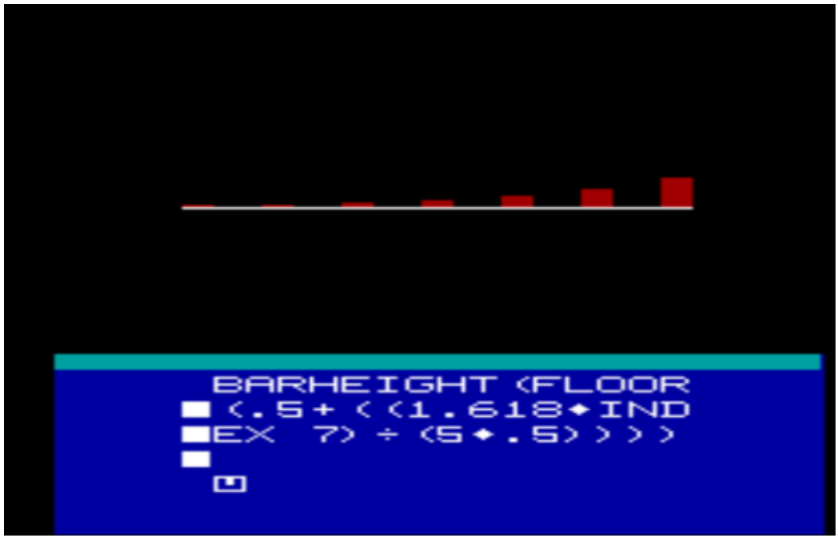
APL/S
FLOOR(.5+((1.61
8◆INDEX 5)÷(5◆.
5)))

8◆INDEX 5) ÷ (5◆.
5)))
1 1 2 3 5


FLOO ⟨0.5+⟨⟨PHI=
⟨⟨Z=5⋄0.5⟩+1⟩×0
.5⟩⋄INDE 16⟩÷Z⟩
1 1 2 3 5 8 13 2
1 34 55 89 144
233 377 610 987

```
FLOOR (.5+((PHI=((Z=5*.5) +1) × .5) * INDEX 5) ÷ Z)
```

The formula for the Fibonacci sequence, using Phi, as a one-liner.

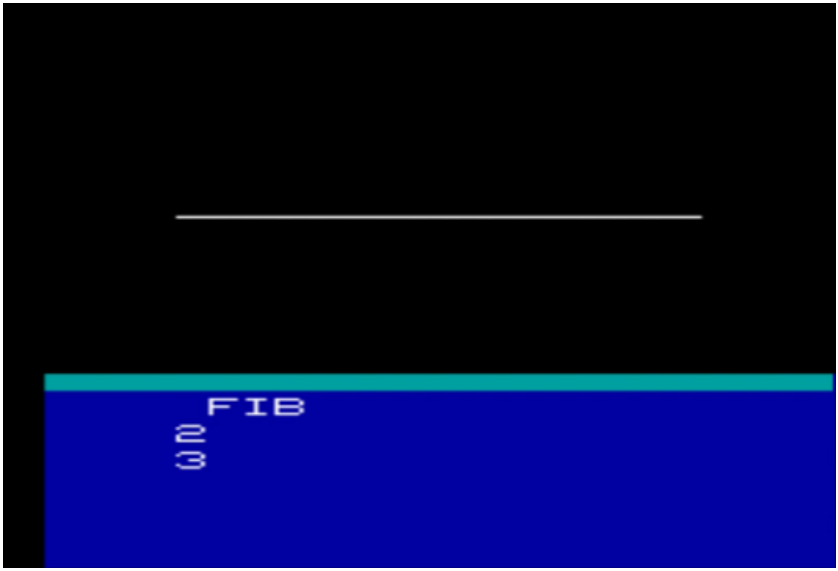


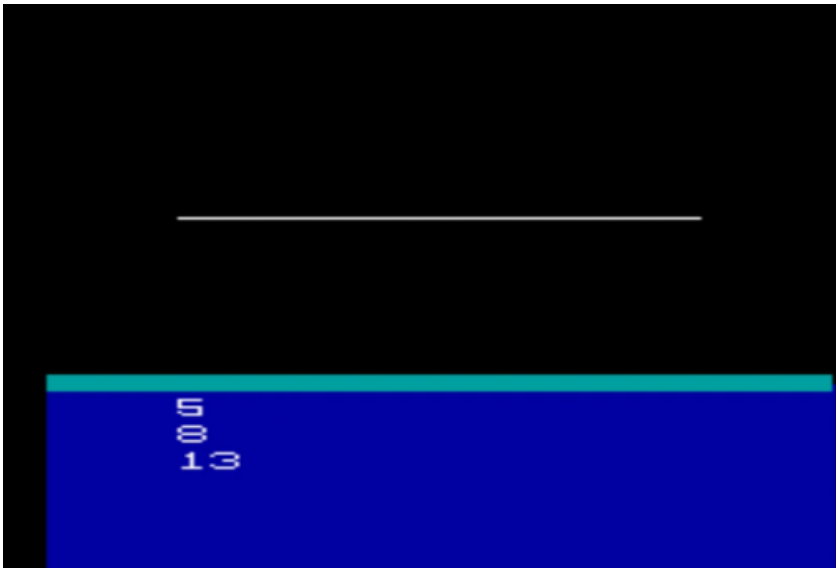
Using the barchart to graph the sequence.

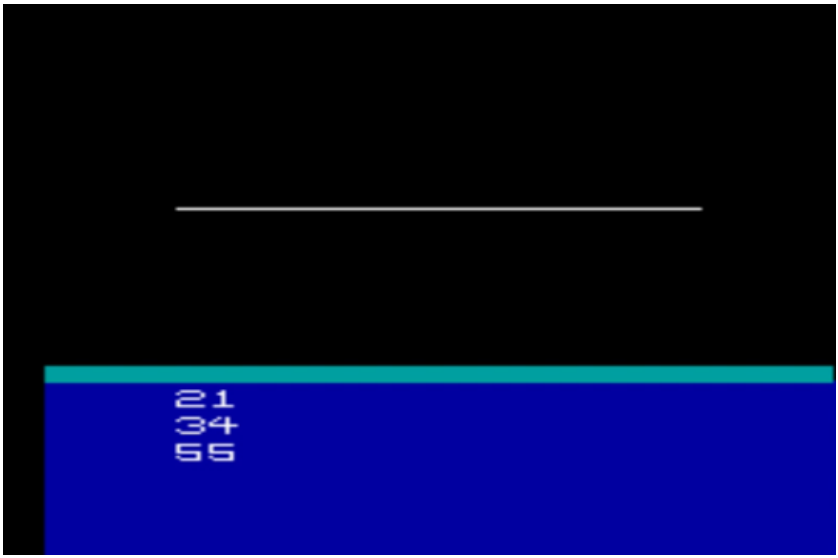
Add a little color. See the APL MOOC to see this done in a modern APL.

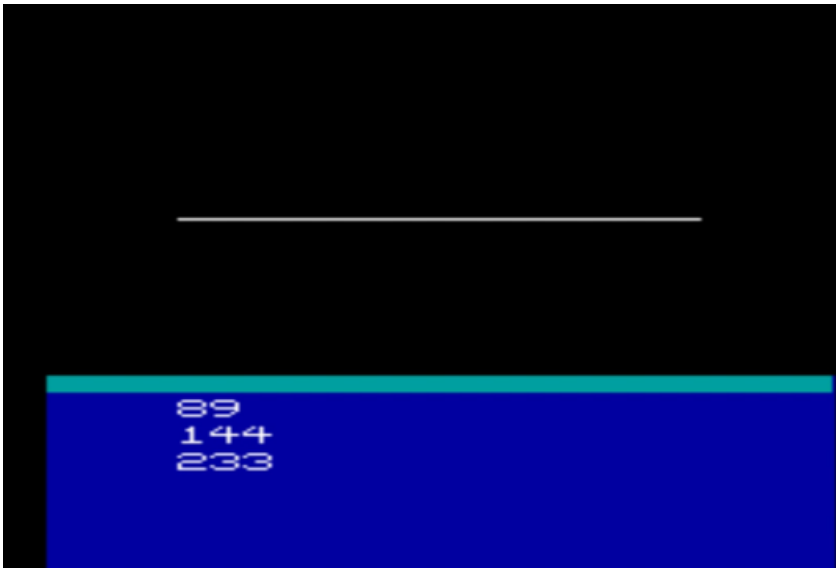
```
X=Y=1

PROG FIB
Z=X+Y
Z
X=Y
Y=Z
FIB
ENDP
```

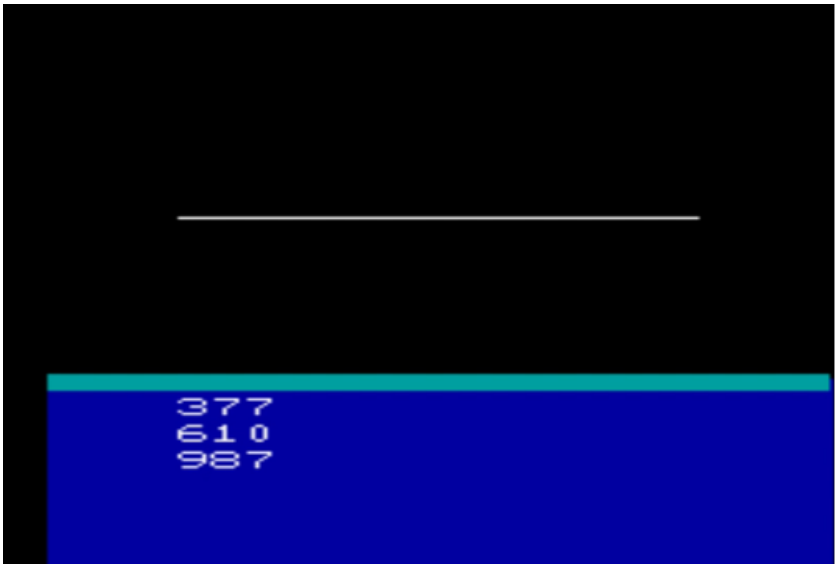
A short recursive program to list the sequence.

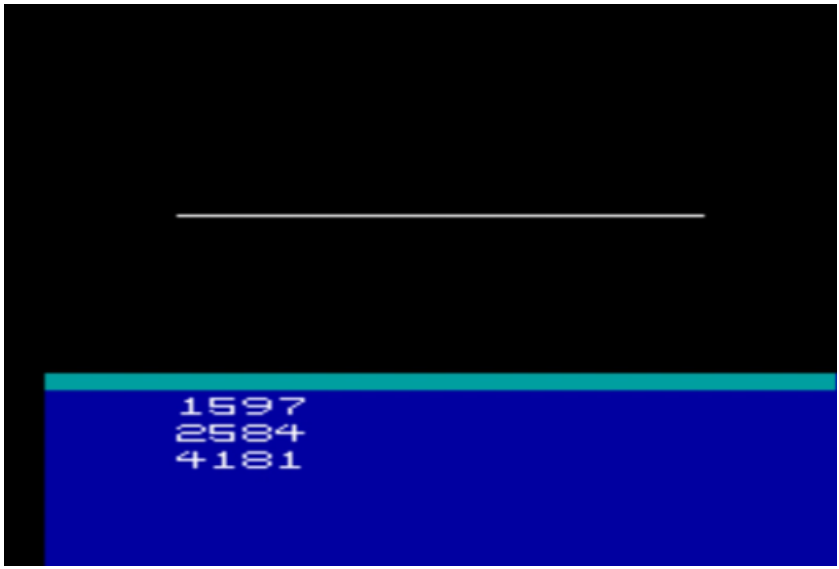



5
8
13


21
34
55


89
144
233

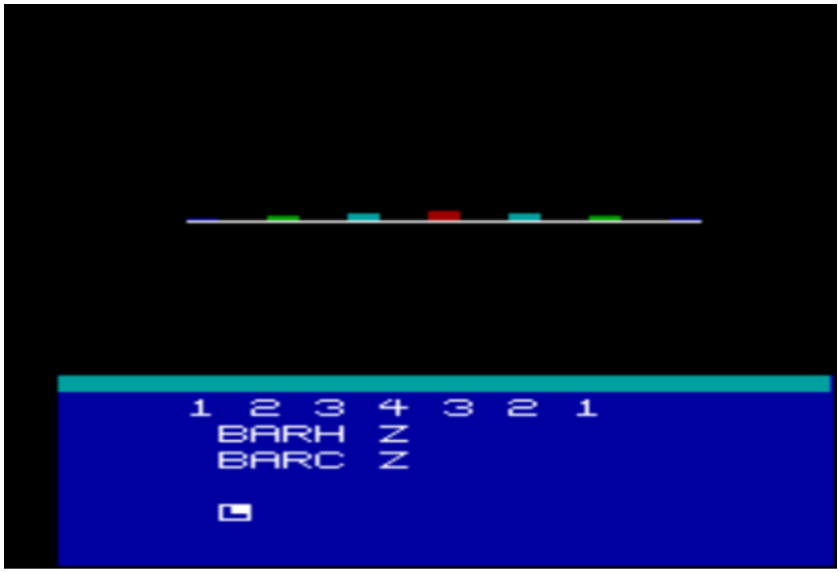

377
610
987



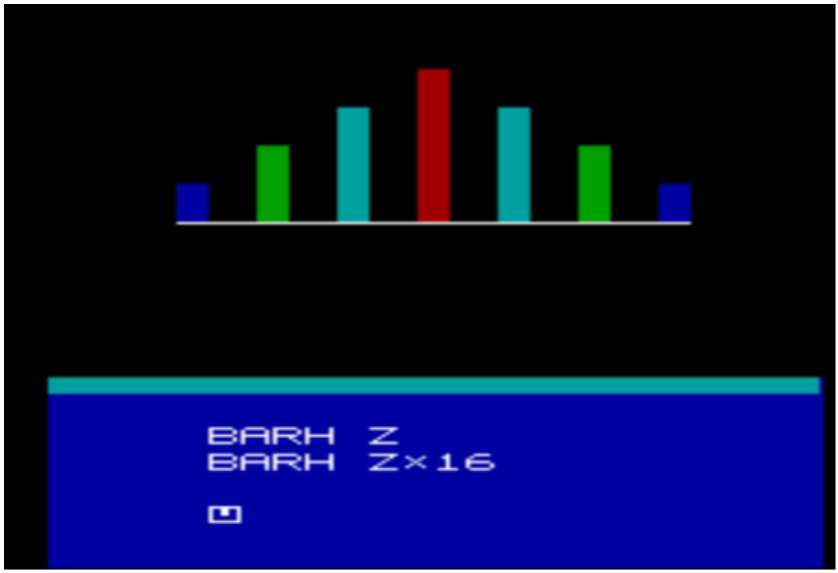
Stack overflow ends the program.

Pyramid chart

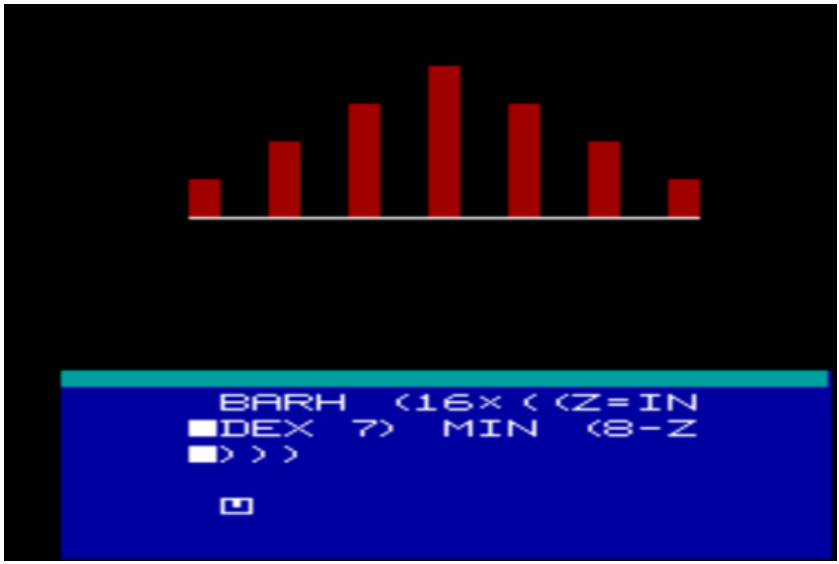
Z=Z MIN(8-Z)
Z
1 2 3 4 3 2 1



A small pyramid.



Scale it with multiplication.



As a one-liner.

Recursive Factorial Fun

In APL/S, recursive functions must account for all variables being global. Outside the program, set X and Y to 1, and assign the value whose factorial you want to N.

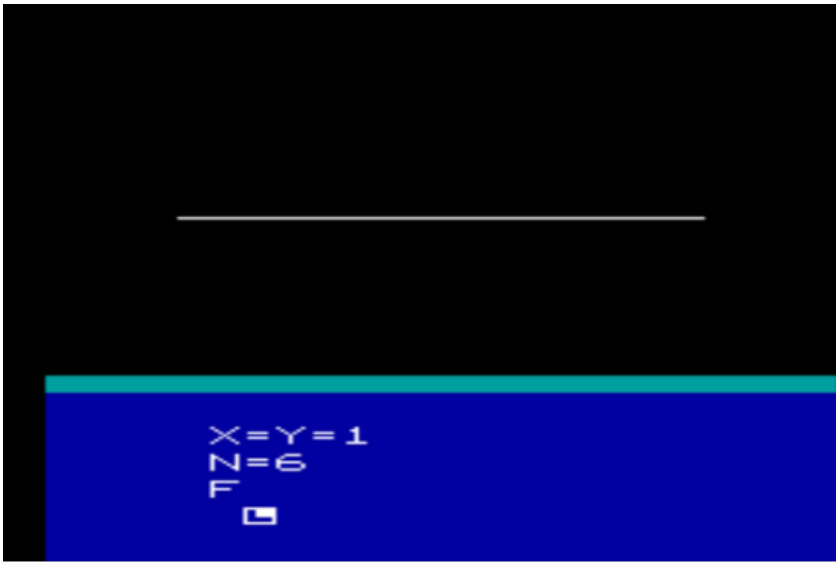
```
X=Y=1
N=10

PROG F
IF X LE N
THEN
    Y=Y×X
    X=X+1
    F
ENDI
ENDP
```

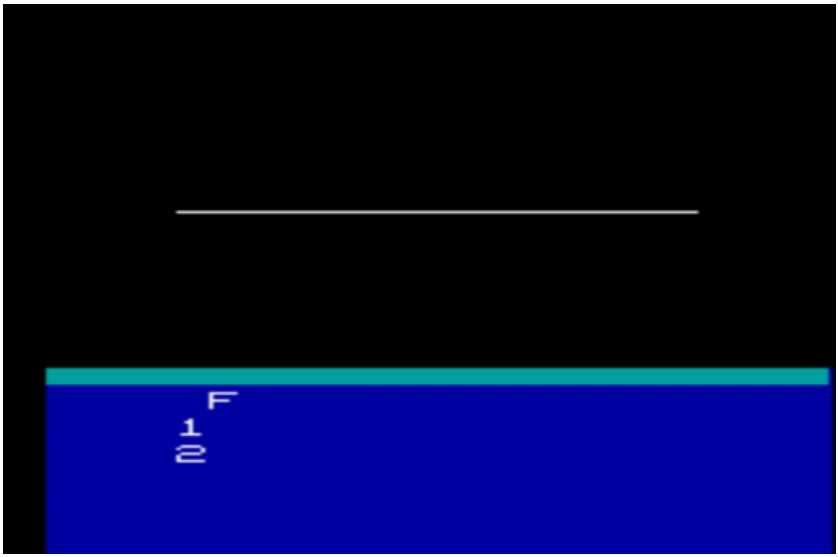
X=Y=1
N=10
F
Y

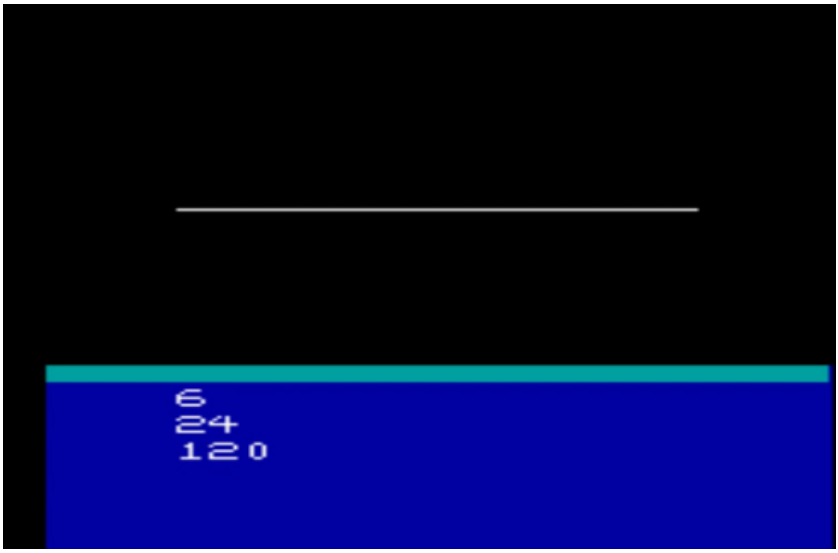
F
Y
362880

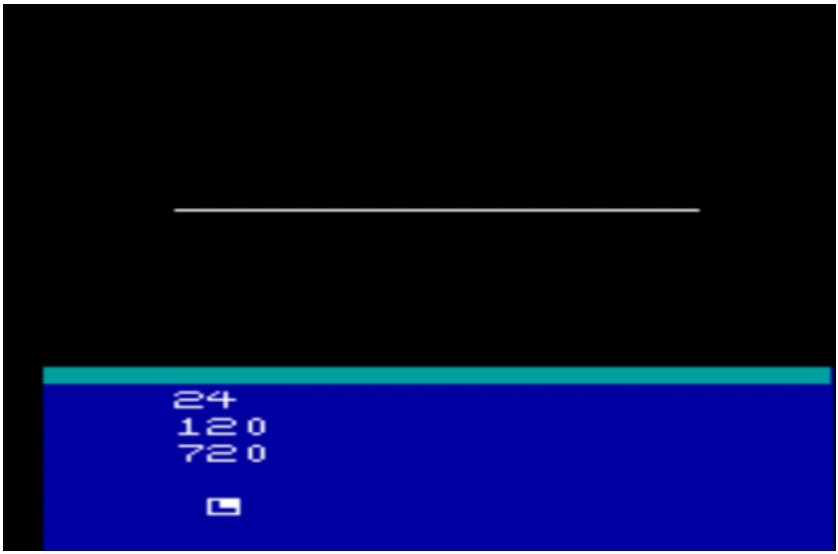
```
X=Y=1
N=6

PROG F
IF X LE N
THEN
    Y=Y×X
    X=X+1
    Y
    F
ENDI
ENDP
```

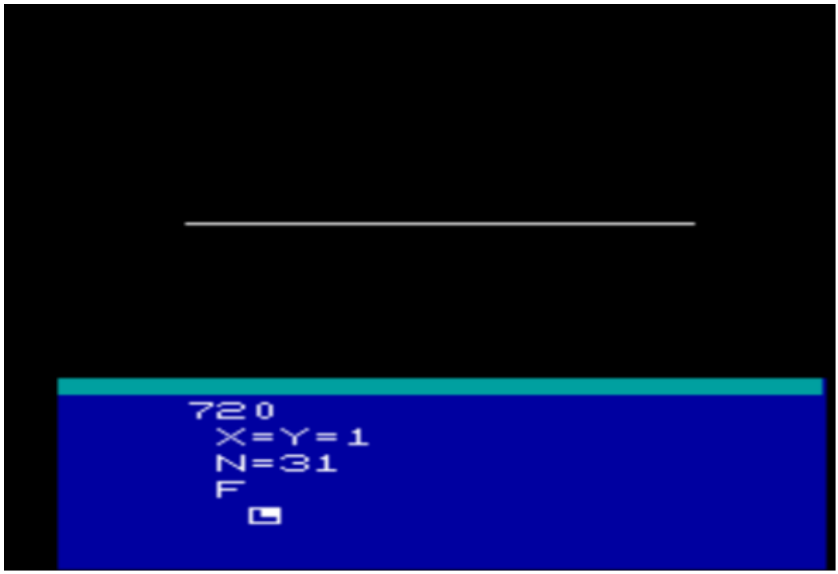
Add a line to print the interim result, Y.


X=Y=1
N=6
F


F
1
2


6
24
120


24
120
720

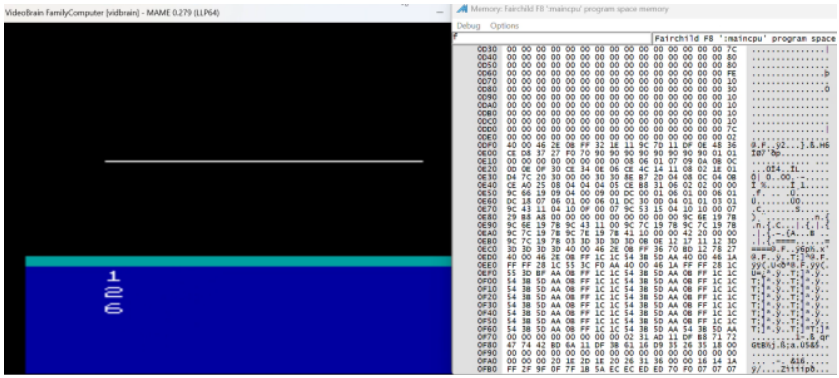

720
X=Y=1
N=31
F

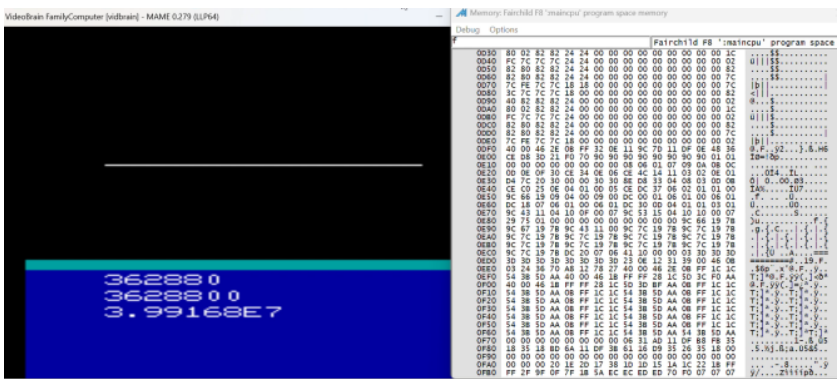
8.841176E30
2.652528E32
8.222838E33

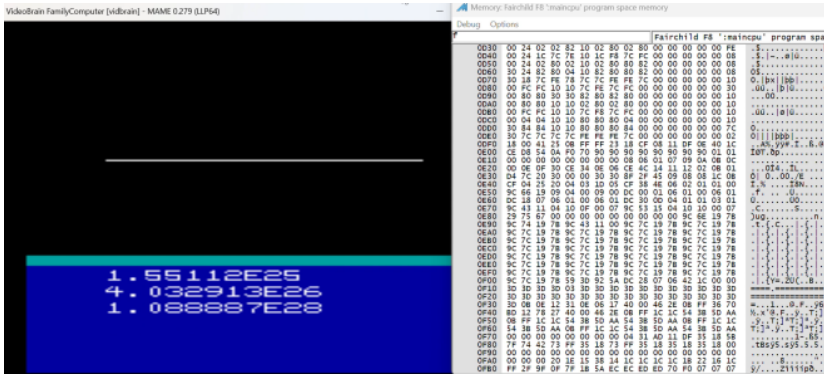
8.222838E33
X=Y=1
N=32
F

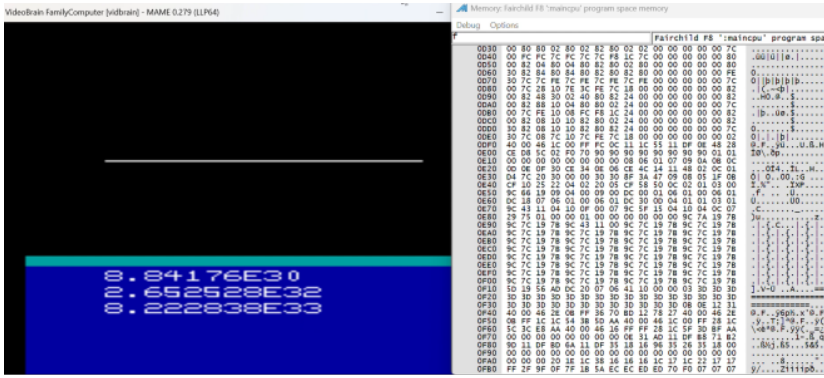
Capacity 206 is an “Out of stack space” error.

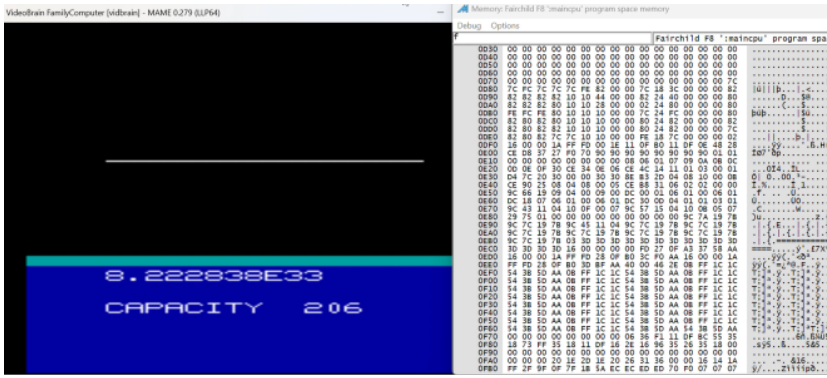
In the MAME debugger, you can see the stack grow as the program runs. Let's overflow the stack again while watching it fill up.






VideoBrain FamilyComputer [vidbrain] - MAME 0.279 (LLP64)
1.55112E25
4.032913E26
1.088887E28
Memory: Fairchild F8 ':maincpu' program space memory
Debug Options
Fairchild F8 ':maincpu' program space


VideoBrain FamilyComputer [vidbrain] - MAME 0.279 (LLP64)
8.84176E30
2.652528E32
8.222838E33
Memory: Fairchild F8 ':maincpu' program space memory
Debug Options
Fairchild F8 ':maincpu' program space



After reaching capacity, the stack overflows.

In APL/S, factorial uses the ! symbol.

:55
1.26964E73

Barheight accepts values over 64 but cuts them off.

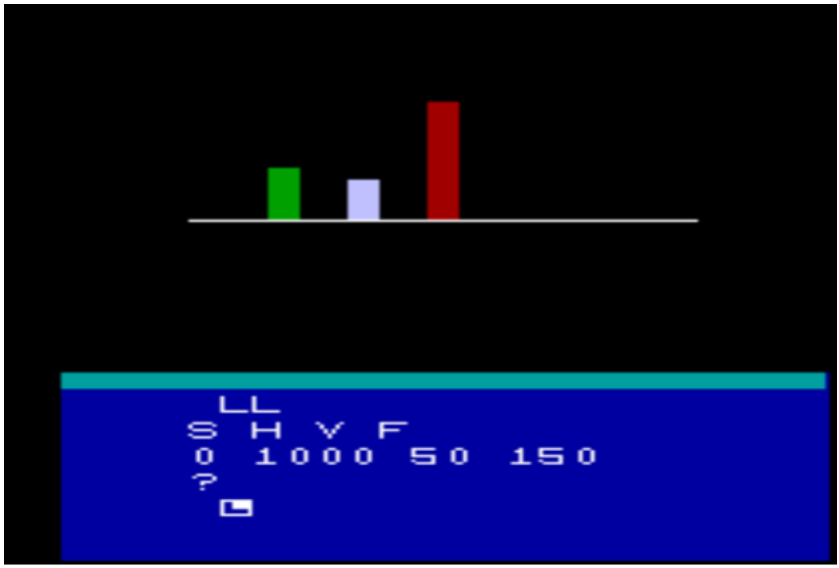
Times reduction can also be used to compute factorials.

Lunar Lander

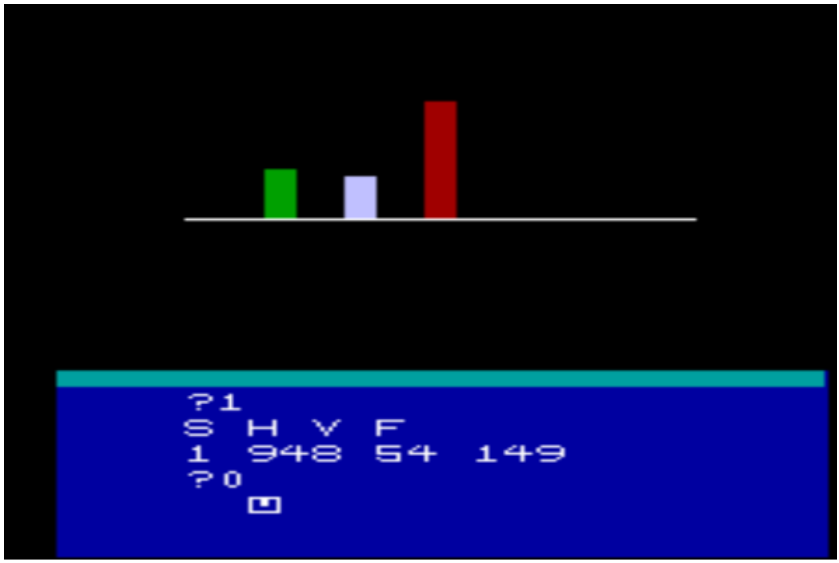
This program is a port of David Ahl's 1973 BASIC game ROCKT1.
Be sure to check the APL/S product brochure for another version of Lunar Lander.

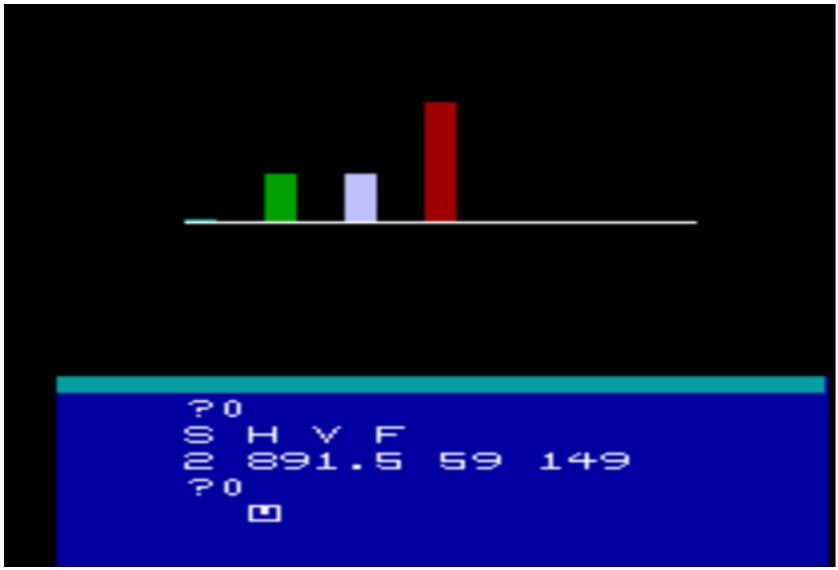
```
PROG LL
T=0
```

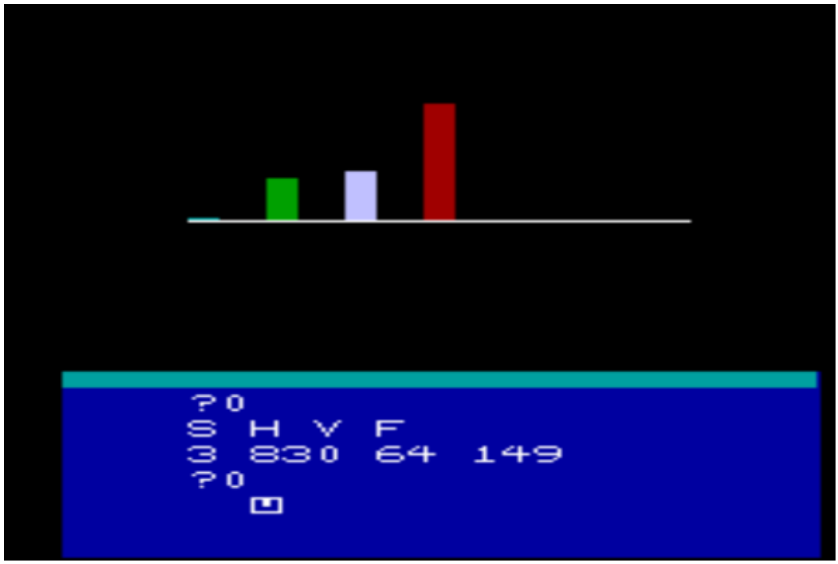
```
PROG LL
T=0
H=1000
V=50
F=150
BARC 3 2 9 4
WHILE H GT 0
DO
   "S H V F"
   T,H,V,F
   BARH T,(H÷15),V,F÷3
   B=KEYB
   IF B GT F
   THEN B=F
   ENDI
   V1=V-B+5
   F=F-B
   H=H-0.5x(V+V1)
   IF H GT 0
   THEN
      T=T+1
      V=V1
      IF F LE 0
      THEN "OUT OF FUEL"
         WHILE H GT 0
         DO
            T,H,V,F
            BARH T,(H÷15),V,F÷3
            V1=V+5
            H=H-0.5x(V+V1)
            T=T+1
            V=V1
         ENDW
      ENDI
   ELSE EXIT
   ENDI
ENDW
"S H V F"
T,H,V,F
"CONTACT"
ENDP
```

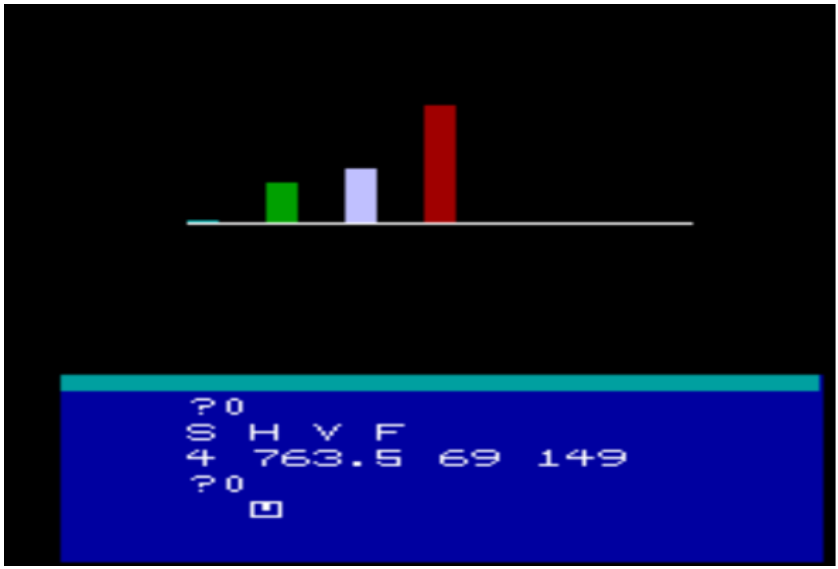


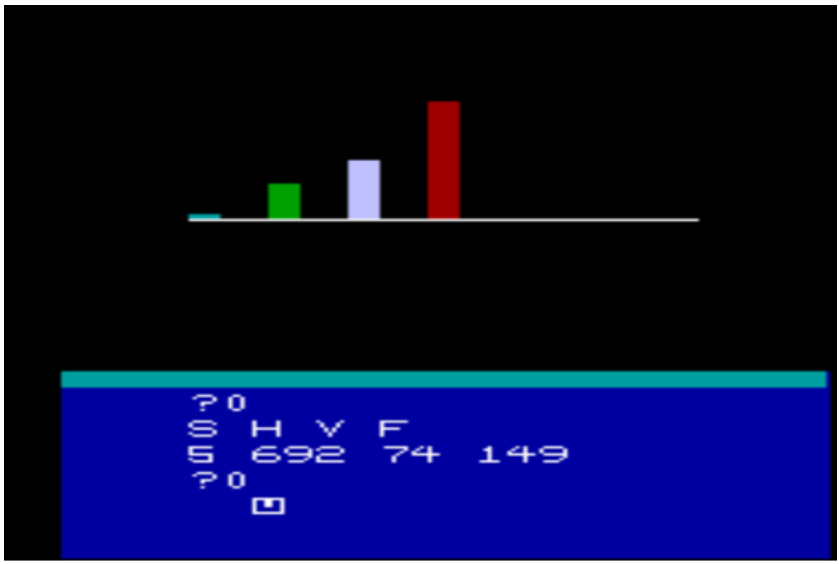
A sample run.

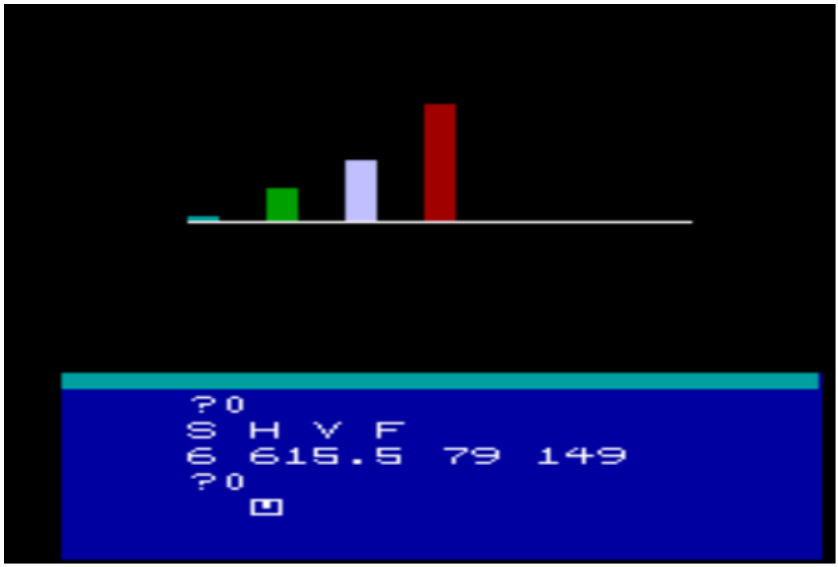


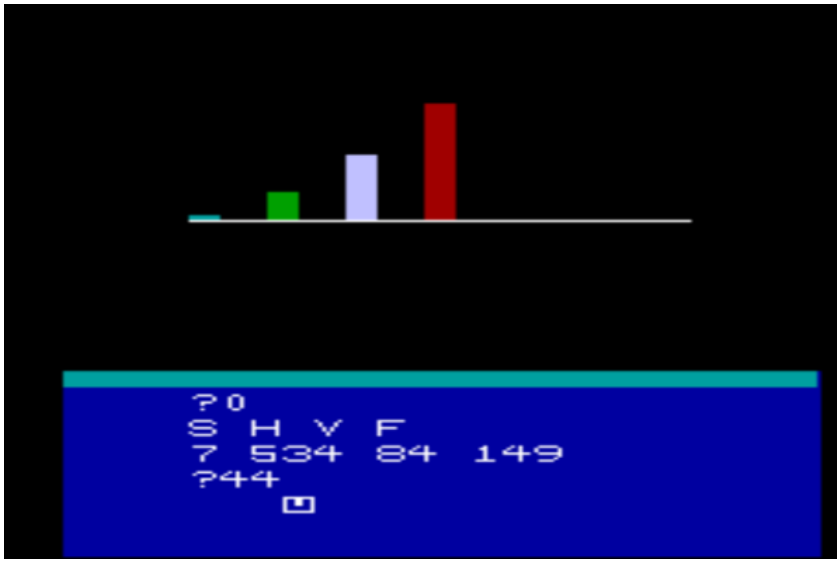

?0
S H V F
2 891.5 59 149
?0

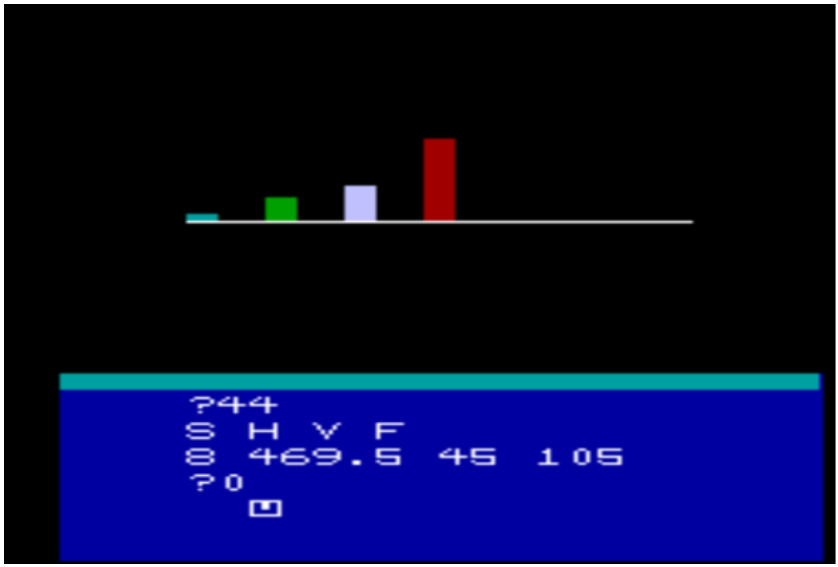

?0
S H V F
3 830 64 149
?0

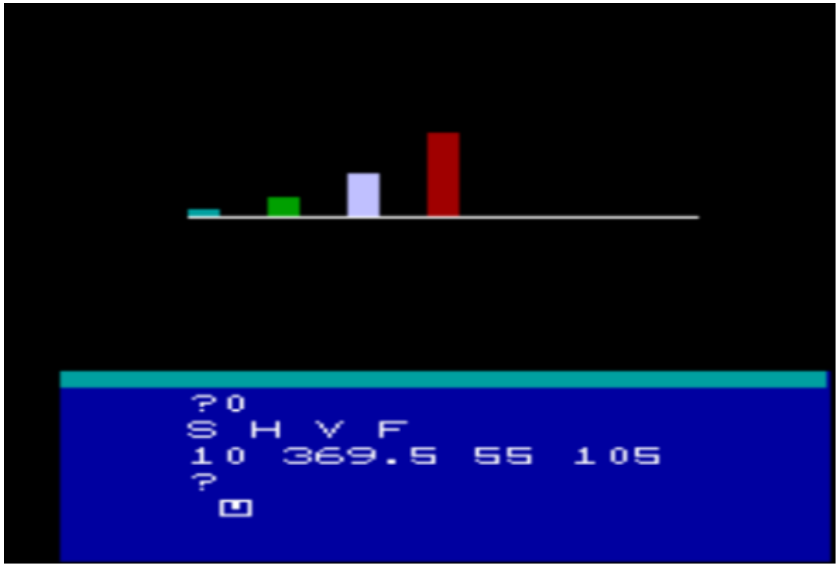

?0
S H V F
4 763.5 69 149
?0

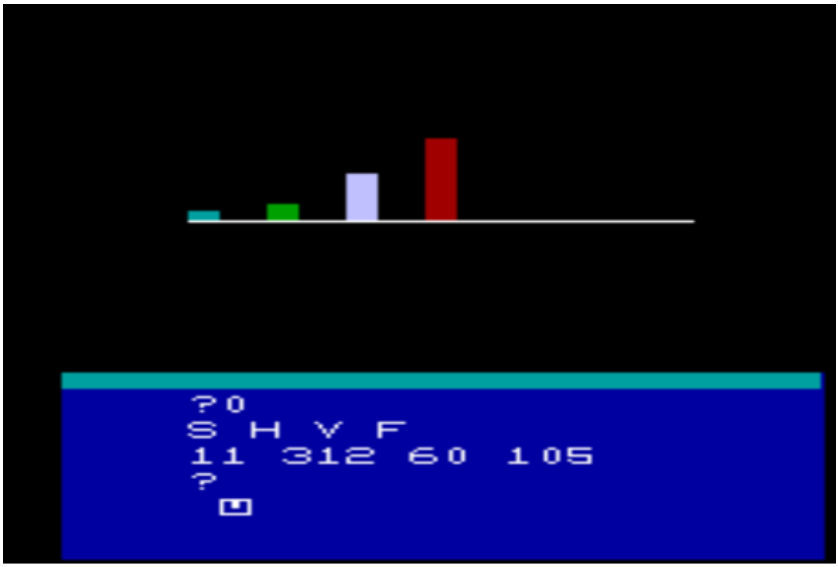

?0
S H V F
5 692 74 149
?0

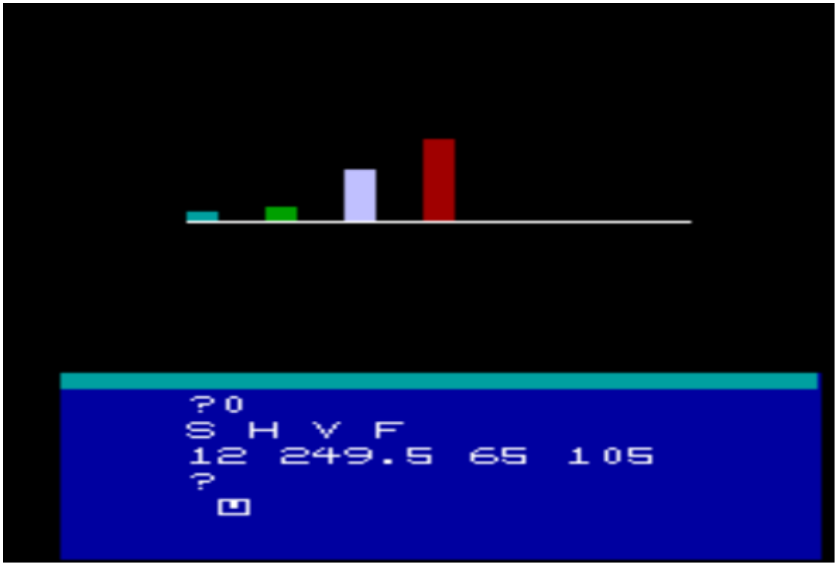

?0
S H V F
6 615.5 79 149
?0

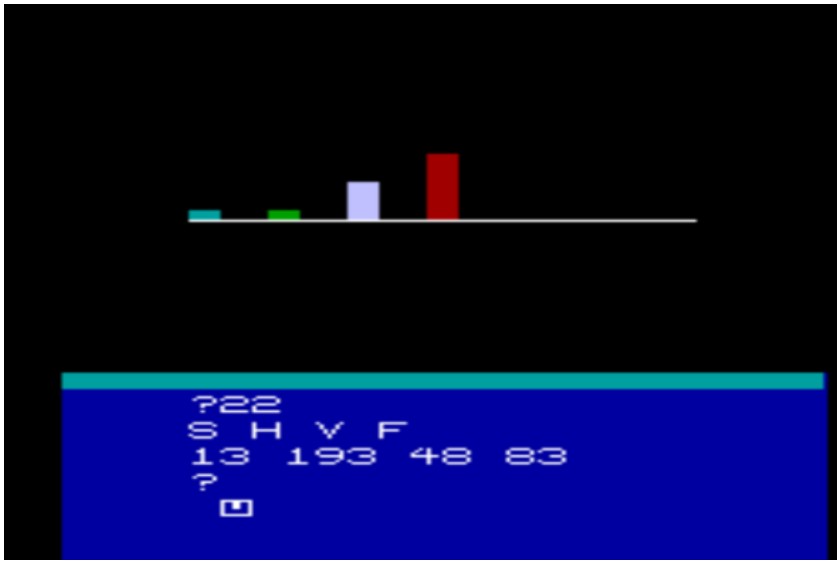

?0
S H V F
7 534 84 149
?44

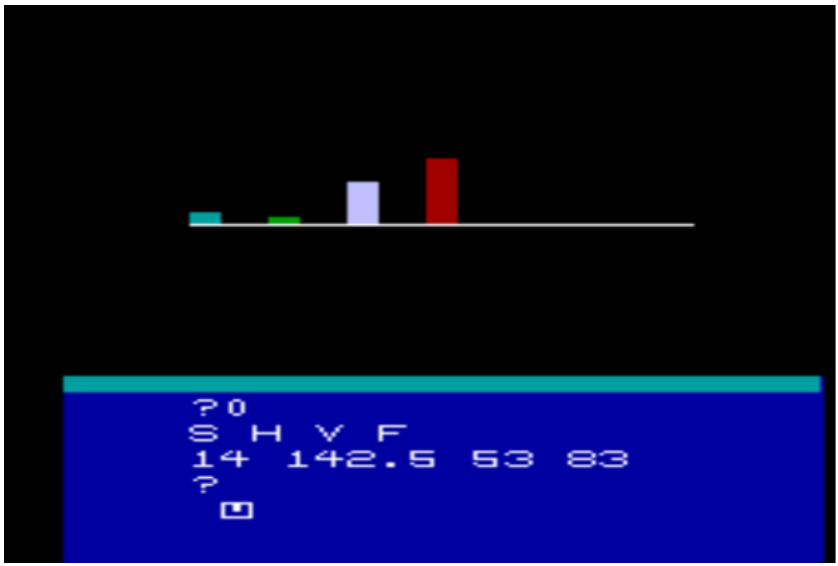

?44
S H V F
8 469.5 45 105
?0

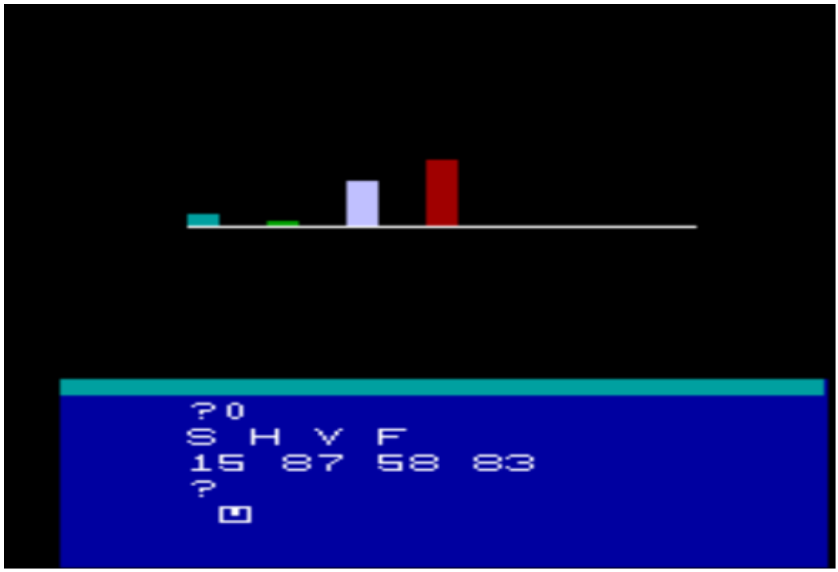

?0
S H V F
10 369.5 55 105
?

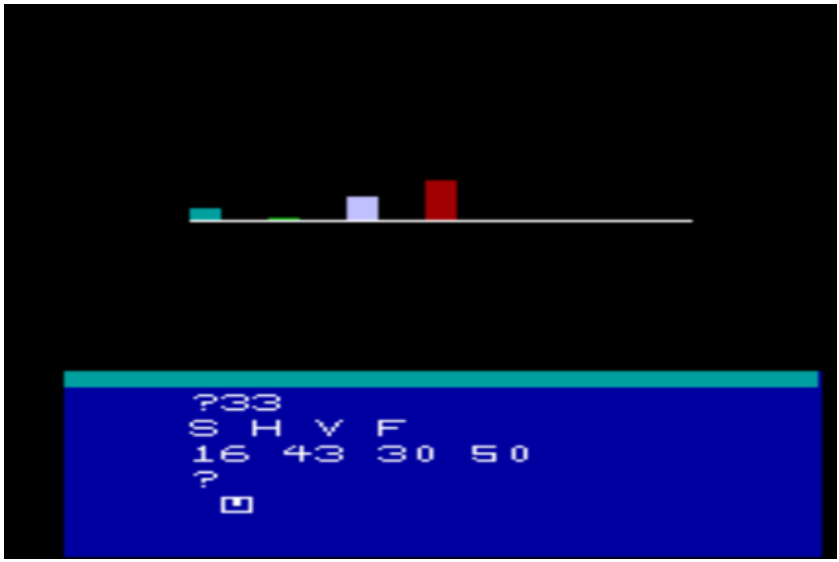

?0
S H V F
11 312 60 105
?

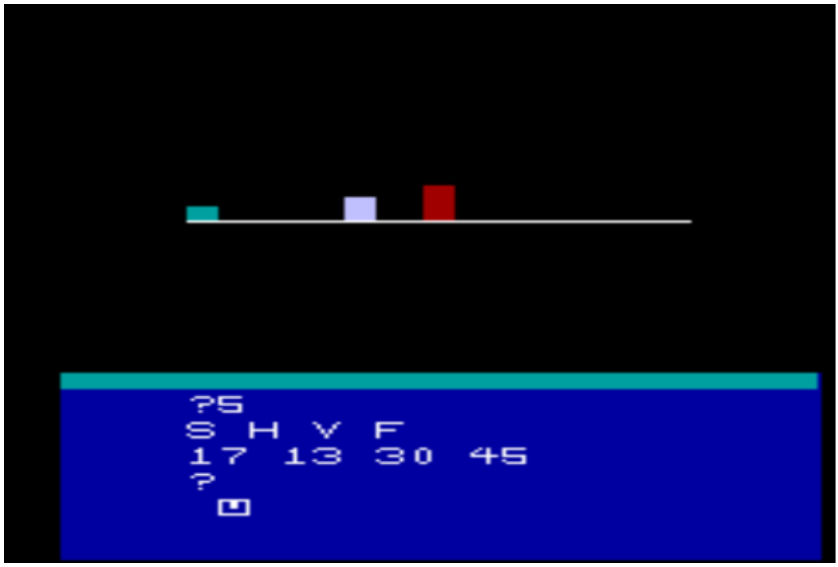

?0
S H V F
12 249.5 65 105
?


?22
S H V F
13 193 48 83
?


?0
S H V F
14 142.5 53 83
?


?0
S H V F
15 87 58 83
?


?33
S H V F
16 43 30 50
?


?5
S H V F
17 13 30 45
?

S H V F
17 -3 30 12
CONTACT

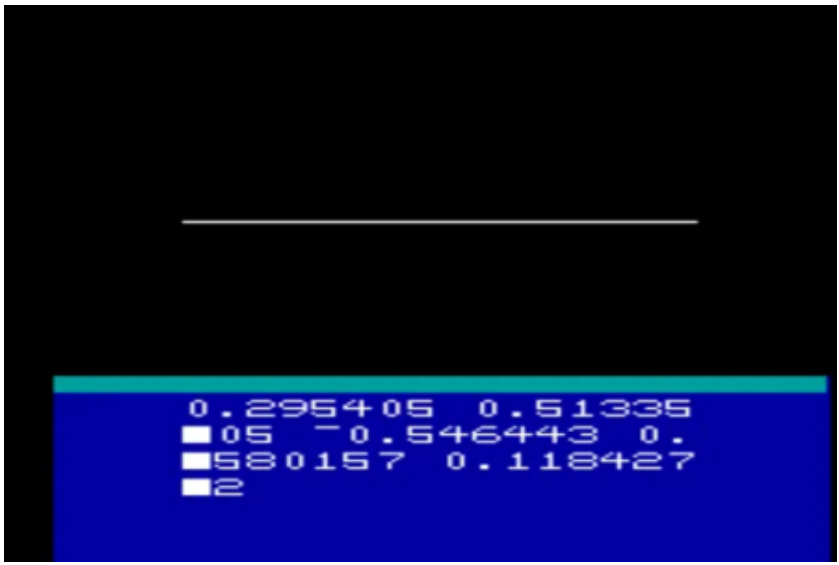
Unit Circle

In his September, 1980 ACM Quote Quad paper **"Uniform sampling on simplices and spheres,"** mathematician Balder Von Hohenbaiken details his APL implementation of an algorithm presented by **Donald Knuth in *The Art of Computer Programming, Volume 2: Seminumerical Algorithms***. Knuth shows how to use the polar method for generating normalized random deviates, developed by **Box, Muller, and Marsagli**. We will use Von Hohenbaiken's technique to generate and display unit circles using APL/S.

```
G←ARRAY 7

PROG UNI
BARC INDEX 7
X←INDEX 0
D←2
WHILE  D  GT  SIZE  X
DO
   WHILE 1
   DO
       V← -1+2*31
       V← -1+((RAND(2×V,(2×V)))÷V)
       IF  1 GT (S←+/(V*2))
       THEN
          EXIT
       ENDI
   ENDW
   X←X,V×(((-2×LOG S)÷S)*.5)
ENDW
IF 1 EQ  (D MOD 2)  // odd dimensions 3, 5, 7 etcetera
THEN
   M←ARRAY D
   MI←1
   WHILE MI LE D
   DO
       M(MI) ← X(MI)
       MI←MI+1
   ENDW
   X←M
ENDI
X←X÷(+/(X*2)*.5)
X
IF D EQ 2
THEN
   I←CEIL (3.5+(X(1)×3))
   H←CEIL (X(2)×64)
   G(I) ← H
   BARH G
ENDI
ENDP
```

In APL/S, all variables are globals. Before running the program, first prepare a global array to store the bar graph height values. As the program is run multiple times, the bar graph will accumulate coordinates.

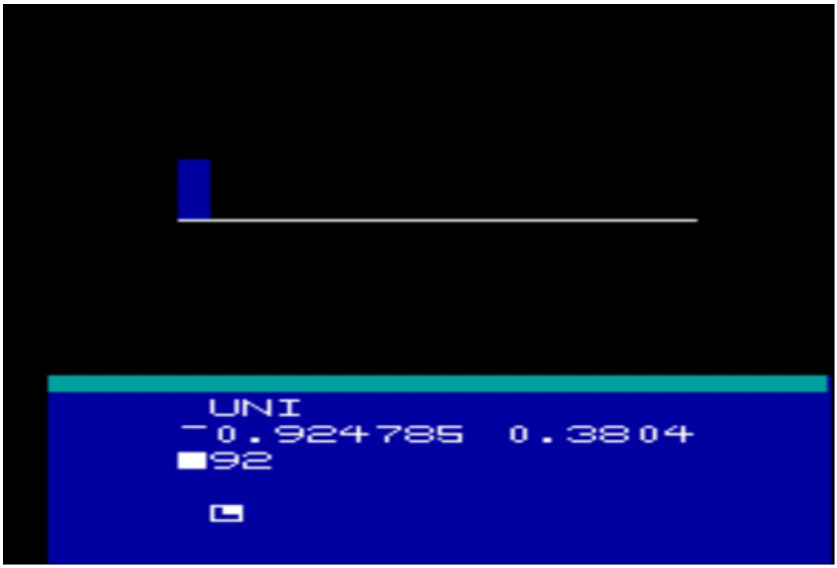
The Box-Muller-Marsagli-Knuth method can generate coordinates for any dimension, although for our unit circle visualization we will only use D = 2. This APL/S implementation can compute coordinates for spheres of higher dimension. Let's try an example run with D = 5:

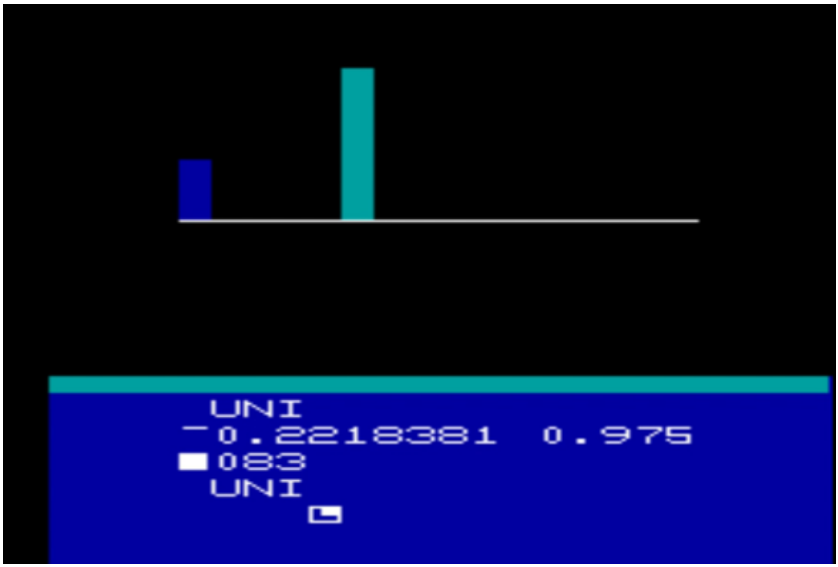


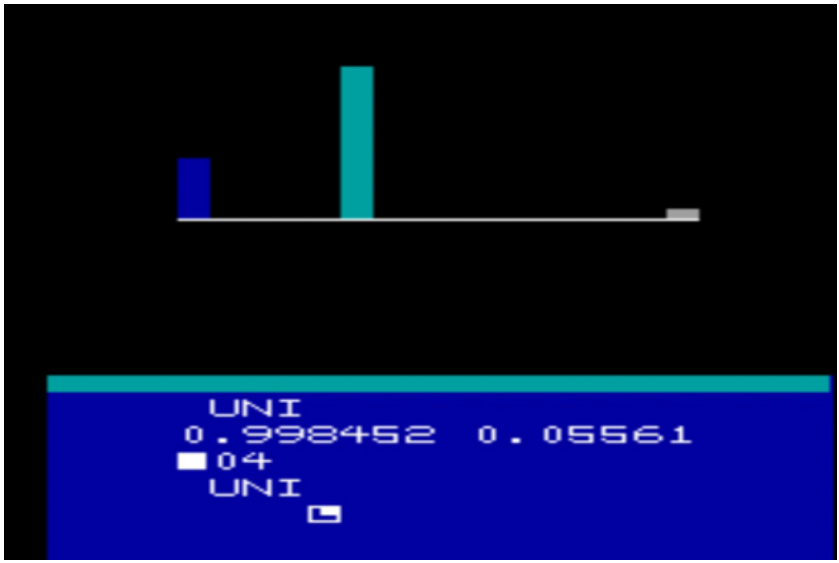
It’s easy to check these coordinates using a calculator:

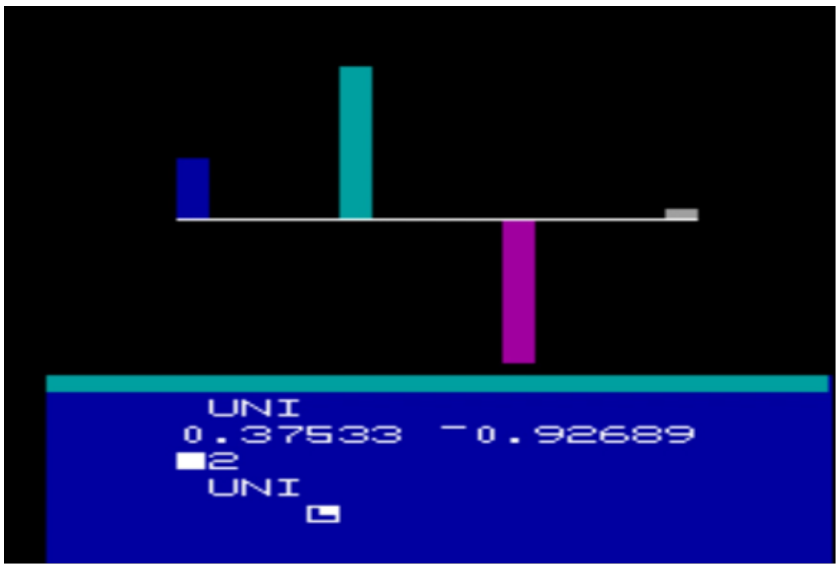
For odd-numbered dimensions, special handling is required since the algorithm generates X[1], X[2] pairs. In the odd cases of D=3, D=5, D=7, ... the line “IF 1 EQ (D MOD 2)” evaluates as true and then, somewhat awkwardly, a new array is created and all the pairs are copied into the new array except for the last pair, where only one member of the pair comes over. This newly created array has “dropped” half a pair, leaving an odd-numbered count of random deviates, done this way because there is no Drop in APL/S. After this, the norm of the resulting vector is computed as usual with X=X ÷ (+/(X*2)*.5).

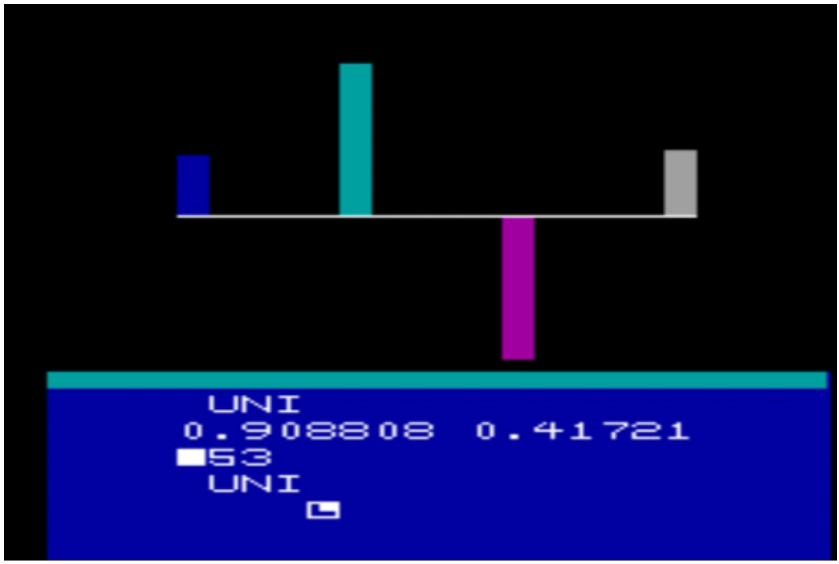
Finally, the program checks if D=2, and if so, scales and shifts the unit circle coordinates to ready them for bar graphing. The VideoBrain’s bar graph facility offers 7 bars, with each bar capable of displaying values from -64 to +64. We can directly visualize the unit circle using this facility:

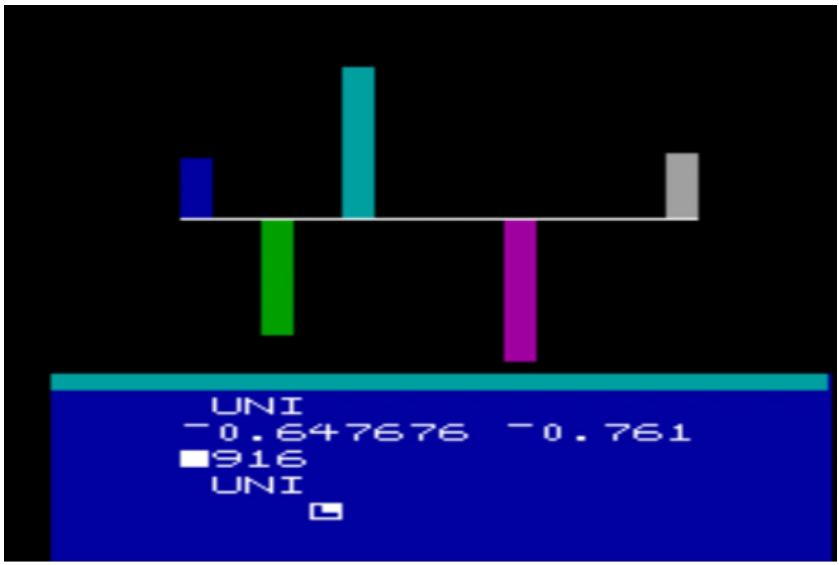

UNI
-0.924785 0.3804
92

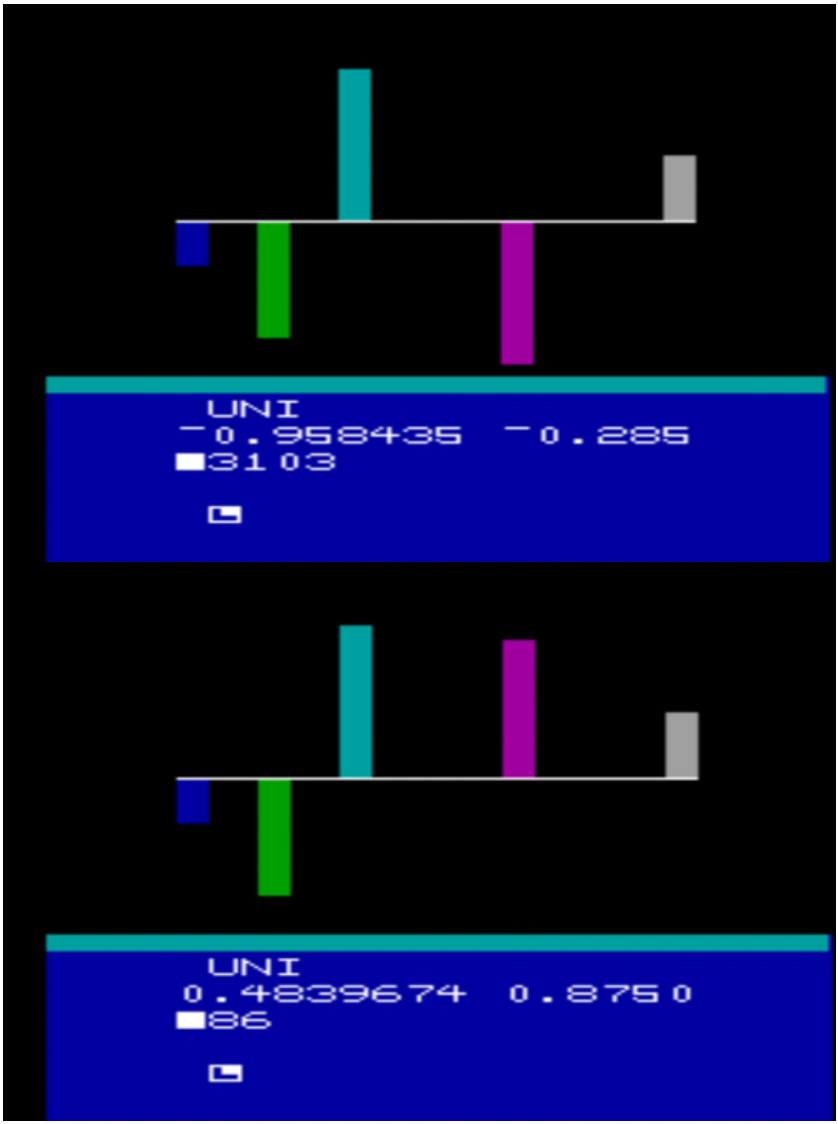

UNI
-0.22183381 0.975
083
UNI

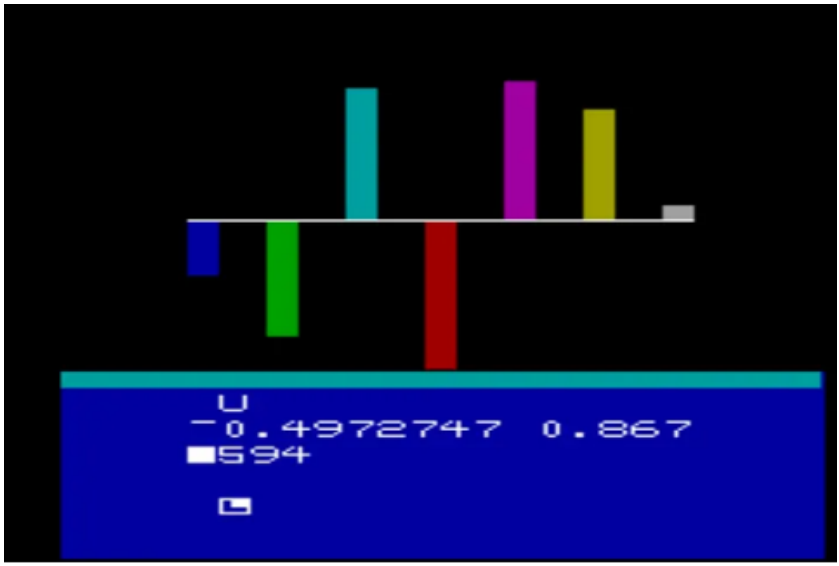

UNI
0.998452 0.05561
04
UNI


UNI
0.37533 -0.92689
2
UNI

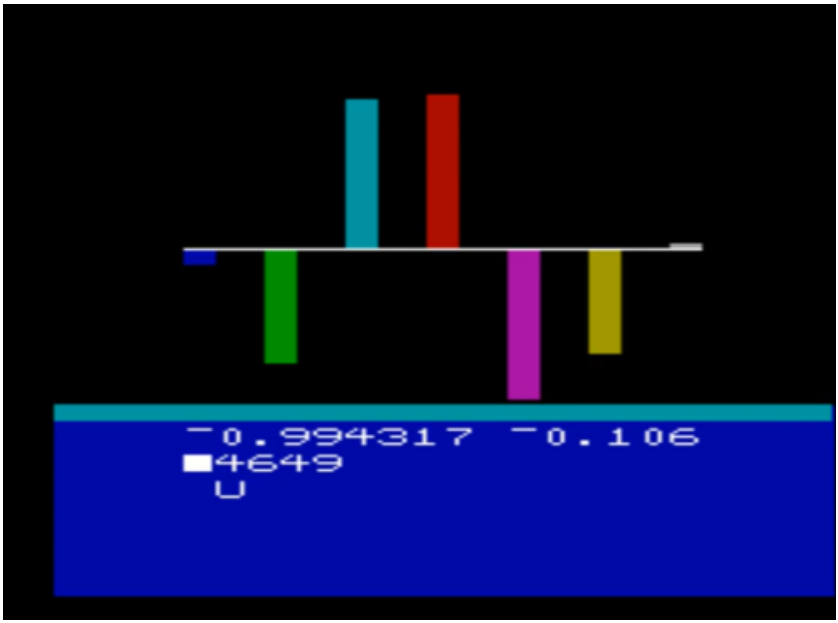

UNI
0.908808 0.41721
53
UNI

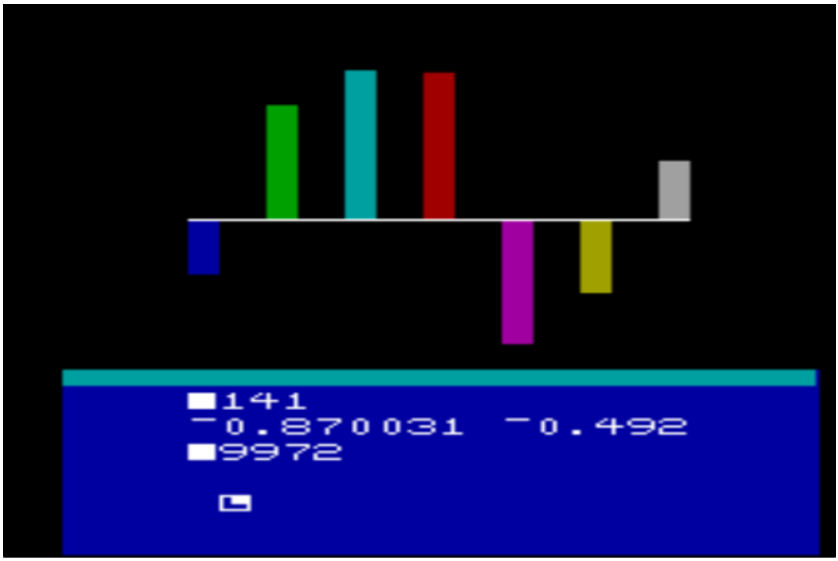

UNI
-0.647676 -0.761
916
UNI


UNI
-0.958435 -0.285
3103
UNI
0.4839674 0.8750
86


U
0.4972747 0.867
594

Another example of showing points being added to a unit circle chart.




141
-0.870031 -0.492
9972

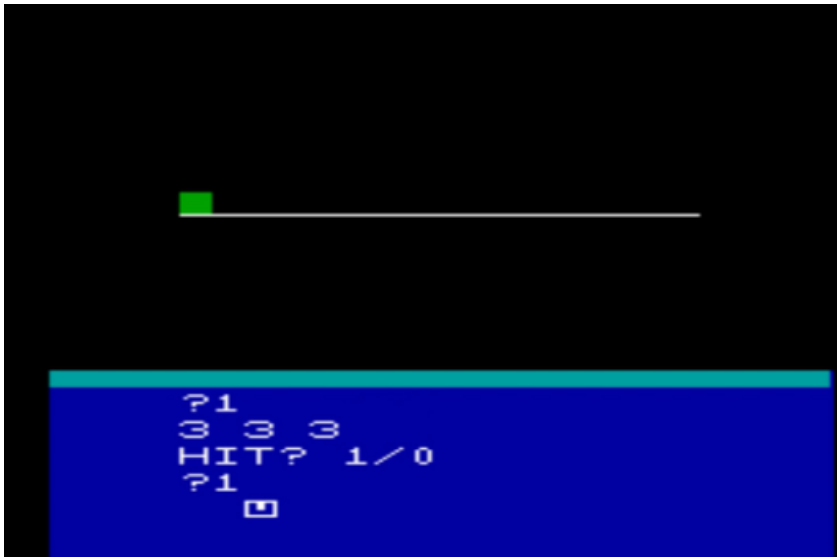
For more, including Commodore 64, C128, and SuperPET implementations, see “Using APL to Visualize the Unit Circle on Vintage 8-bit Computers”

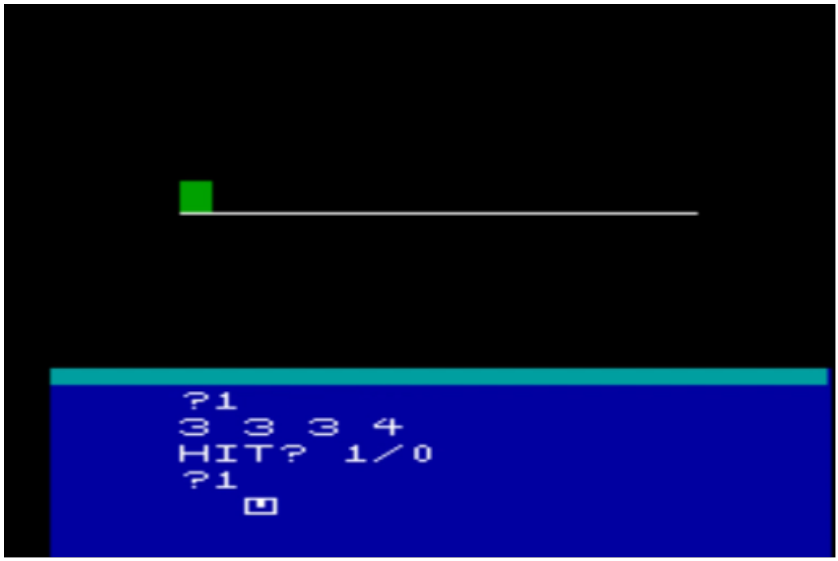
Dicejack

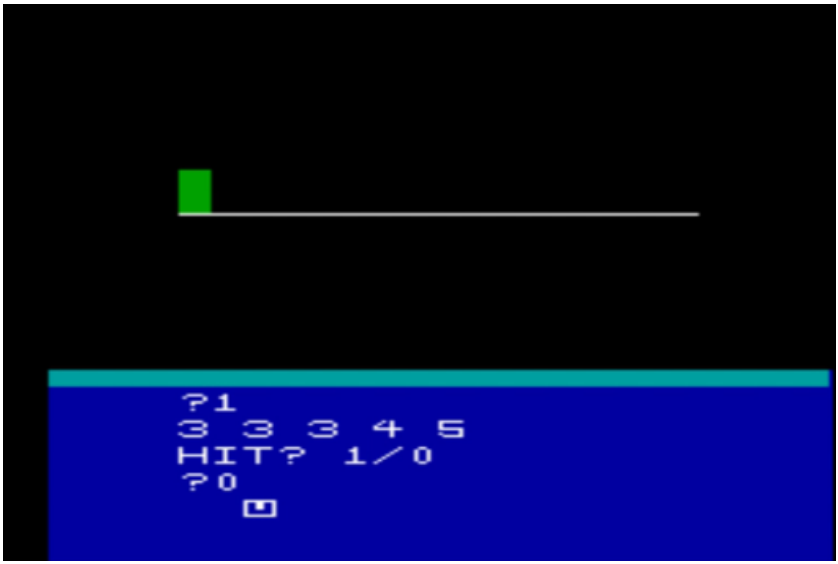
A dice version of Blackjack, similar to the game “Twenty-One” from “The Gateway Guide To The ZX81 And ZX80” by Mark Charlton. The book is part of the Creative Computing material released into the public domain by David Ahl.

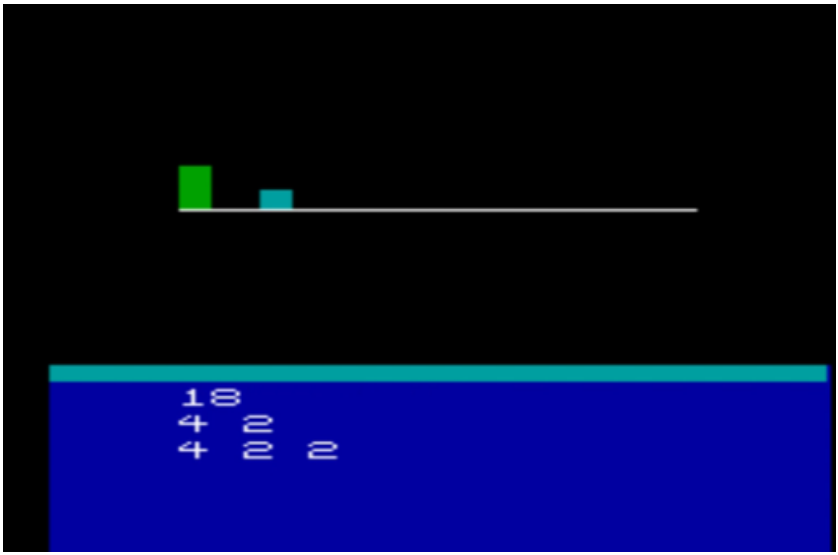
```
PROG D
BARC 2 3
BARH +/(Z=RAND 6 6),0
Z
WHIL 1
DO
   'HIT? 1/0'
   C=KEYB
   IF C EQ 0
   THEN EXIT
   ENDI
   IF C EQ 1
   THEN Z=Z,RAND 6
      Z
      BARH +/Z
      IF +/Z GT 21
      THEN "BUSTED"
         EXIT
      ENDI
   ENDI
ENDW
+/Z
IF +/Z LE 21
THEN
   B=0   //DEALER NOT BUSTED
   X=RAND 6 6
   X
   WHIL (+/Z) GE (+/X)
   DO
      X=X,RAND 6
      X
      BARH +/Z,+/X
      IF +/X GT 21
      THEN "I BUSTED"
         B=1
         EXIT
      ENDI
   ENDW
   +/X
   IF B EQ 1
   THEN "YOU WIN"
   ELSE
      IF +/X GT +/Z
      THEN "I WIN"
      ELSE "YOU WIN"
      ENDI
   ENDI
ENDI
ENDP
```

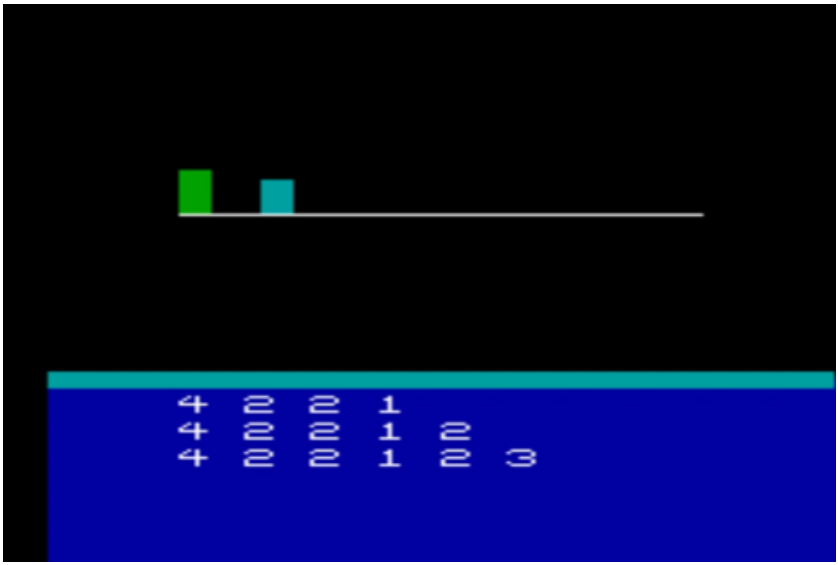
D
3 3
HIT? 1/0
?1

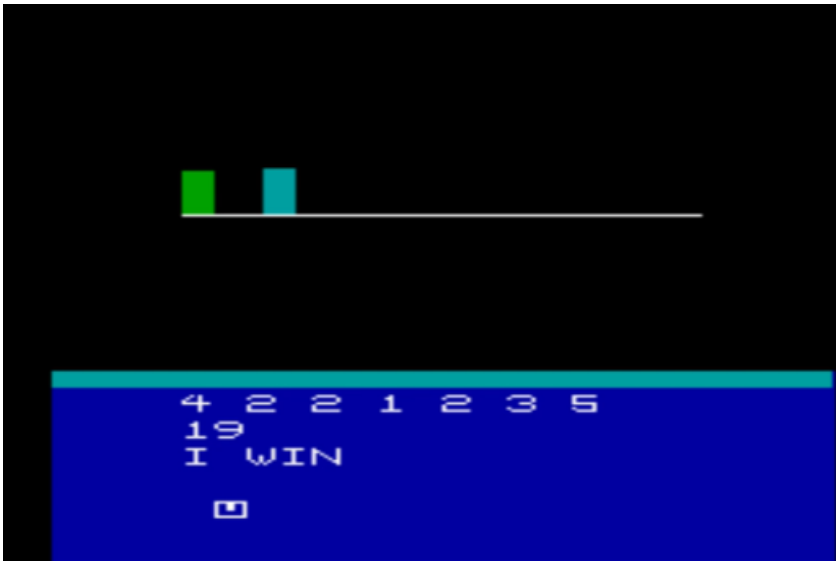

?1
3 3 3
HIT? 1/0
?1


?1
3 3 3 4
HIT? 1/0
?1


?1
3 3 3 4 5
HIT? 1/0
?0

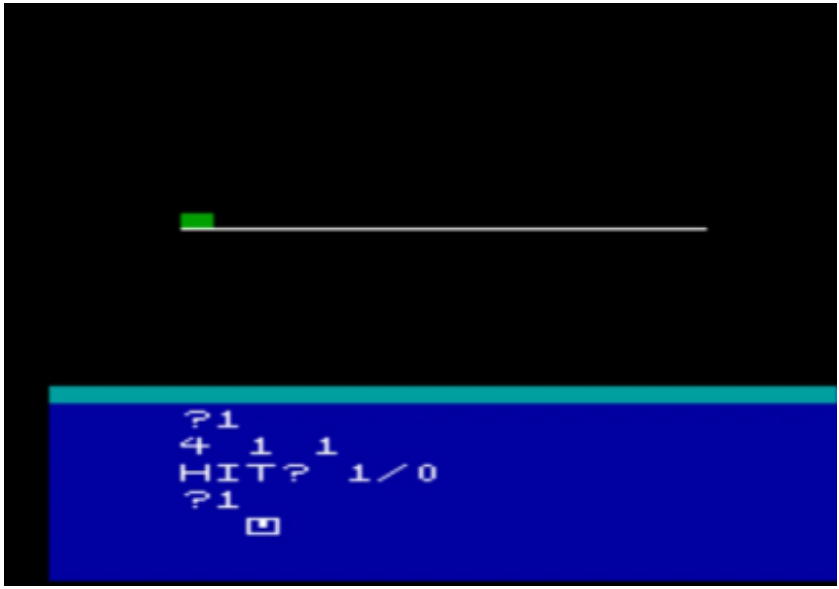

18
4 2
4 2 2

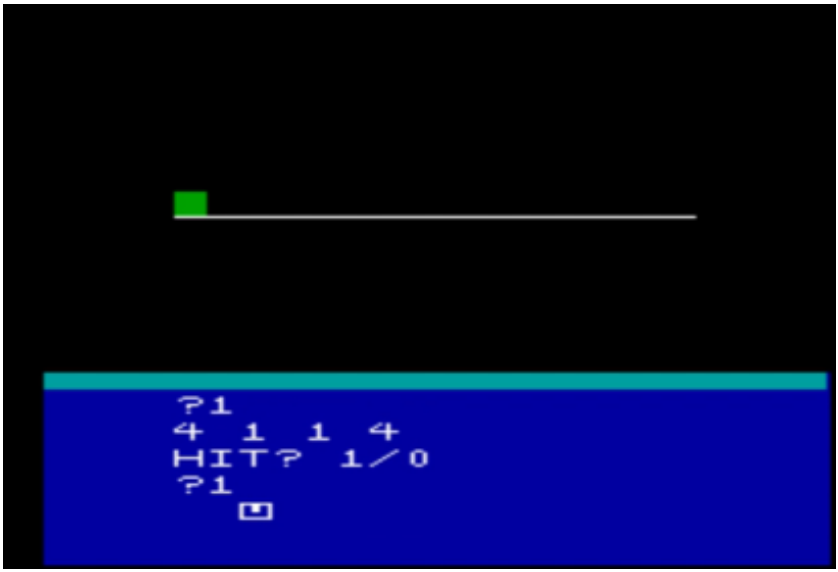

4 2 2 1
4 2 2 1 2
4 2 2 1 2 3

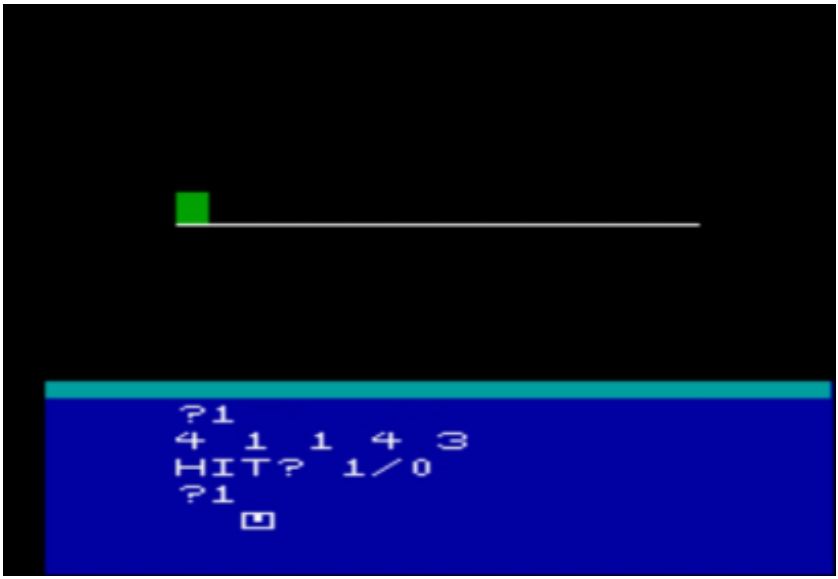

4 2 2 1 2 3 5
19
I WIN

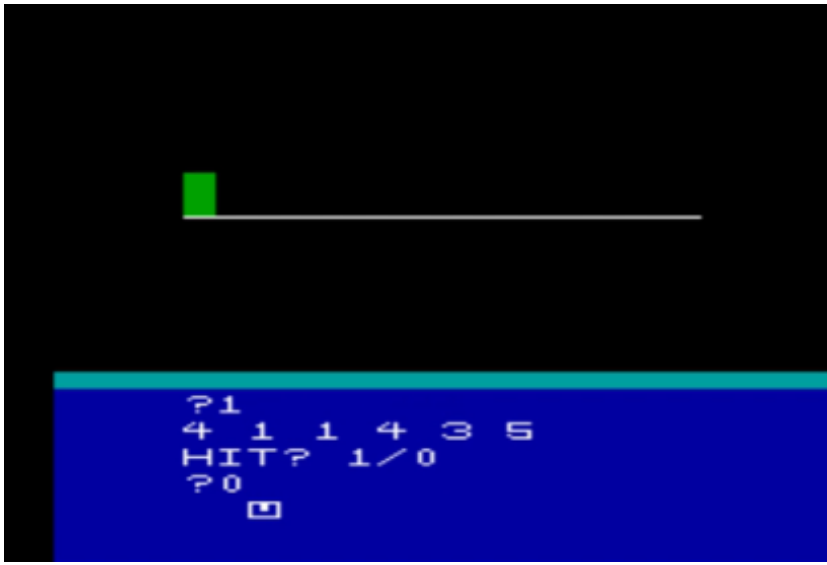
DJ
4 1
HIT? 1/0
?1

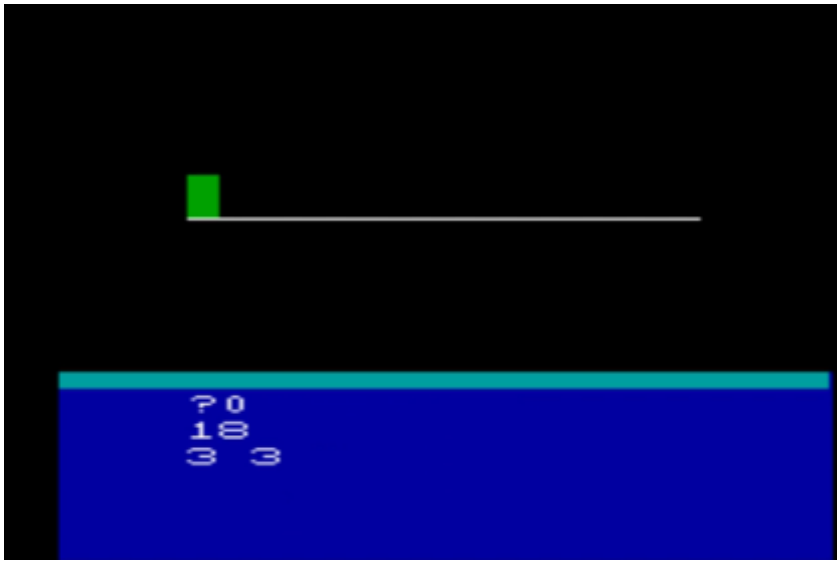
Another game.

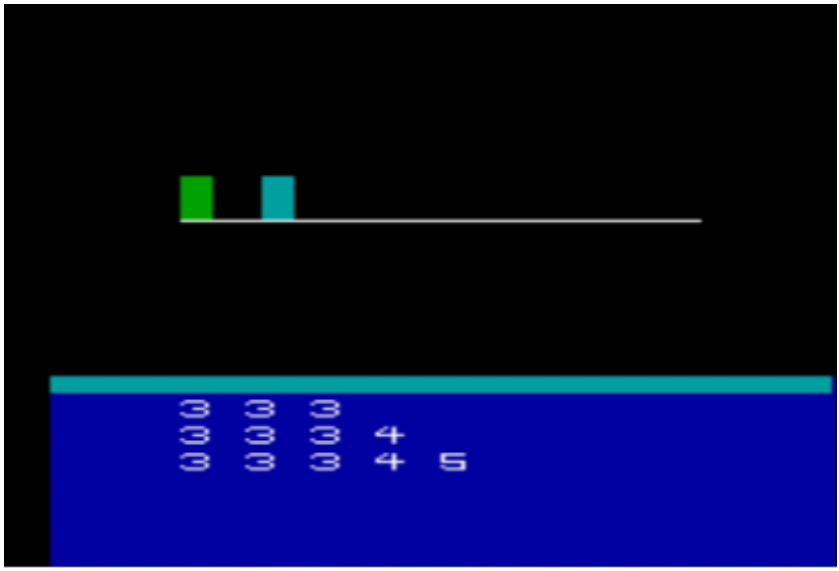



?1
4 1 1 4
HIT? 1/0
?1


?1
4 1 1 4 3
HIT? 1/0
?1


?1
4 1 1 4 3 5
HIT? 1/0
?0


?0
18
3 3


3 3 3
3 3 3 4
3 3 3 4 5

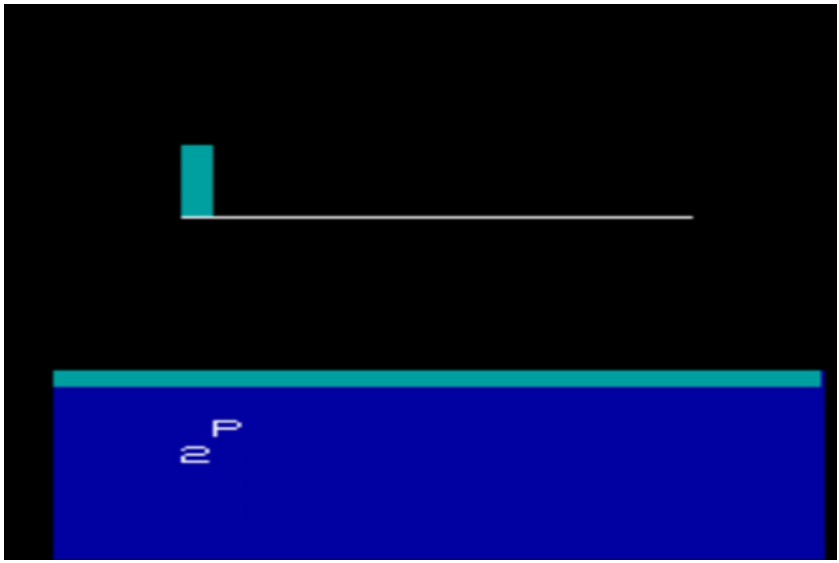
3 3 3 4 5 6
I BUSTED
24

I BUSTED
24
YOU WIN

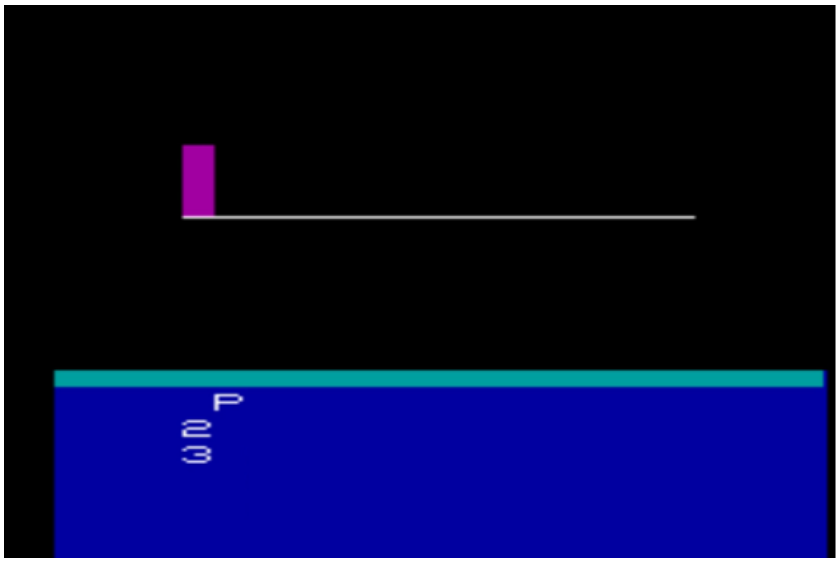
Primes

```
PROG P
BARC 2
BARH 30
2
BARC 3
3
I=4
UPTO=23
WHIL I LE UPTO
DO
   PRIM=1
   IF I MOD 2 EQ 0
   THEN PRIM=0
   ELSE
      J=3
      WHIL J LE (I * .5)
       DO
          IF I MOD J EQ 0
          THEN PRIM=0
             EXIT
          ELSE J=J+2
          ENDI
       ENDW
   ENDI
   BARC 1+(I MOD 7)
   IF PRIM EQ 1
   THEN BARH 30
      I
   ELSE
      BARH -30
   ENDI
   I=I+1
ENDW
ENDP
```

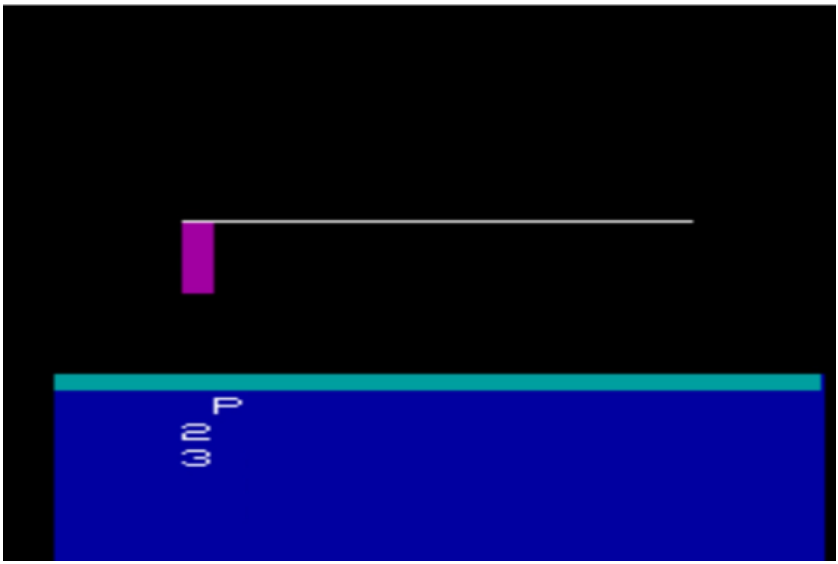
First, consider a program that uses a primality by trial division algorithm.



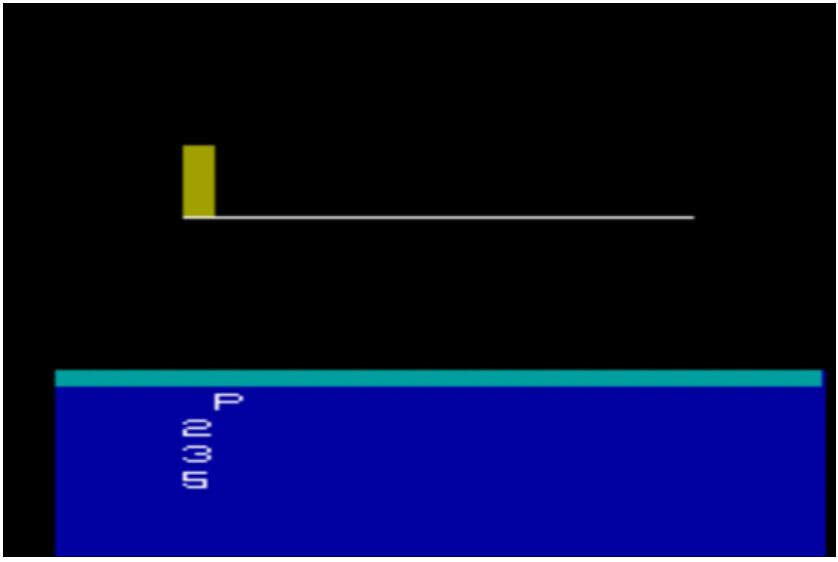
2 is prime, so the flag goes up to indicate a prime was found.



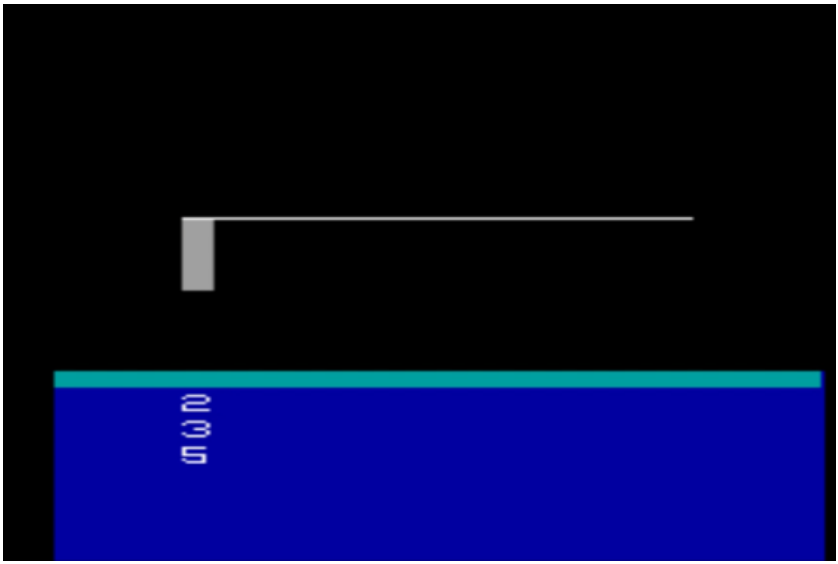
3 is also prime, the flag remains up and shifts color accordingly.



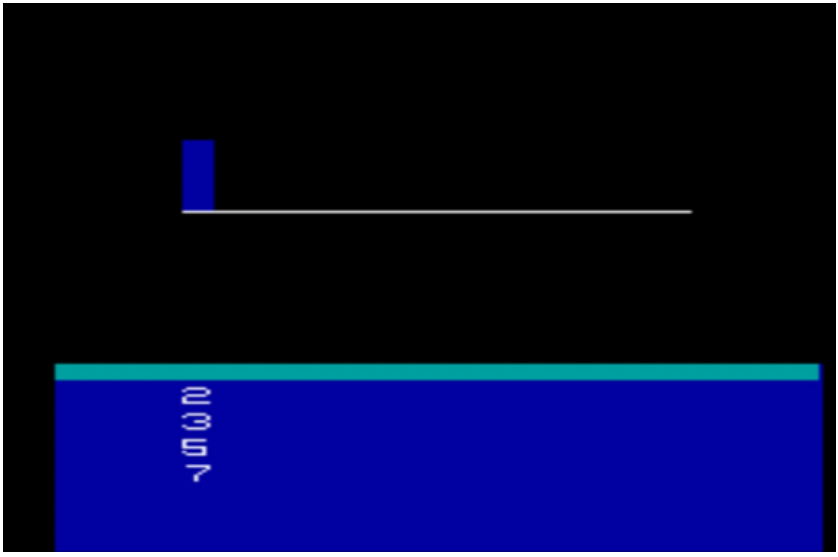
Down for 4.



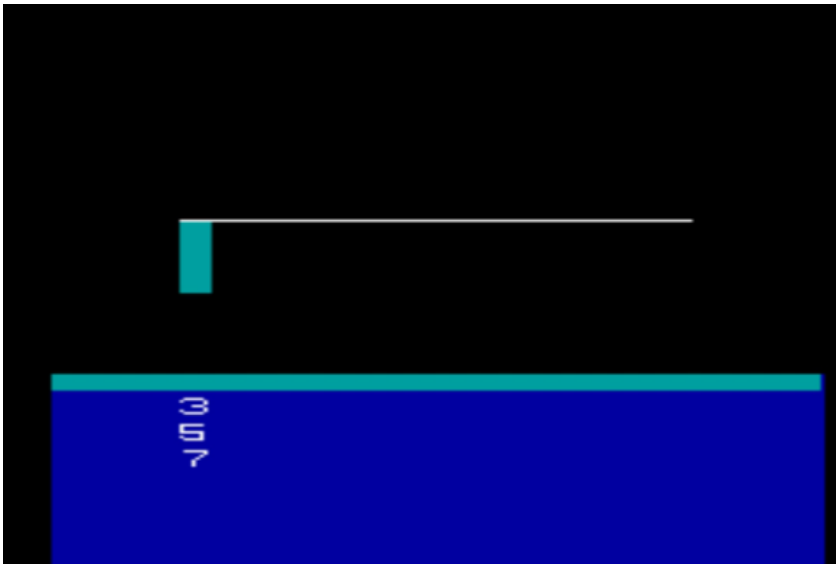
Up for 5.



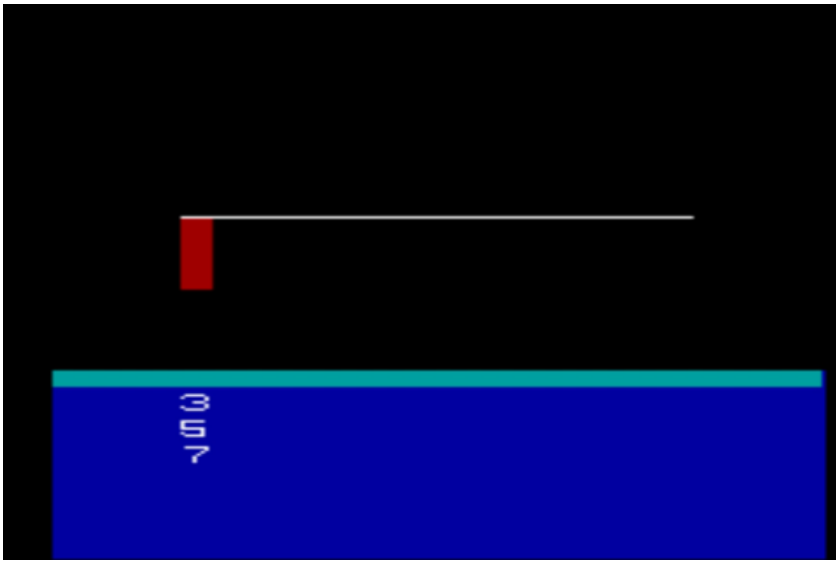
Down for 6.



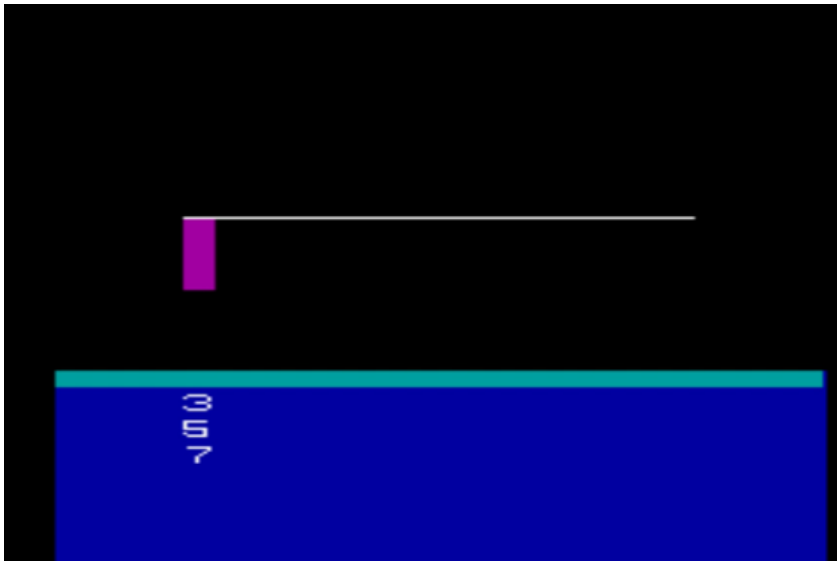
Up for 7.



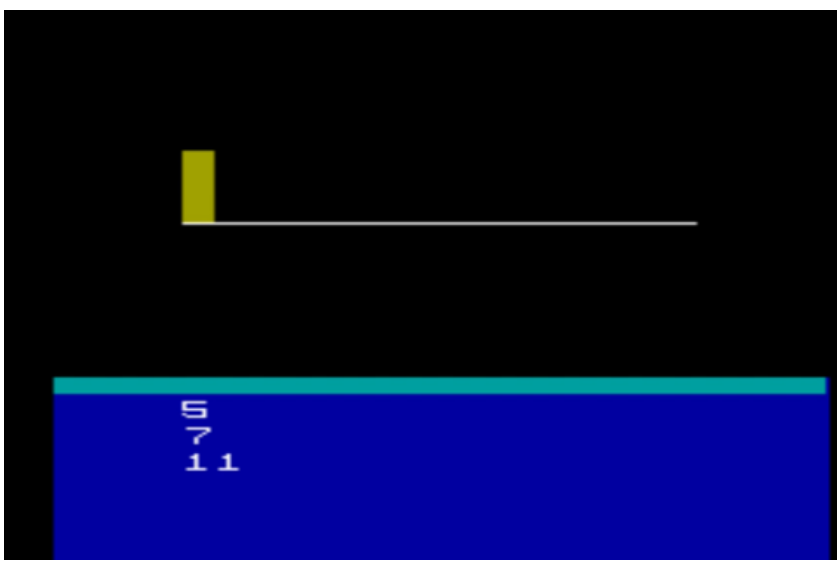
Down for 8.



Down for 9.



Down for 10.



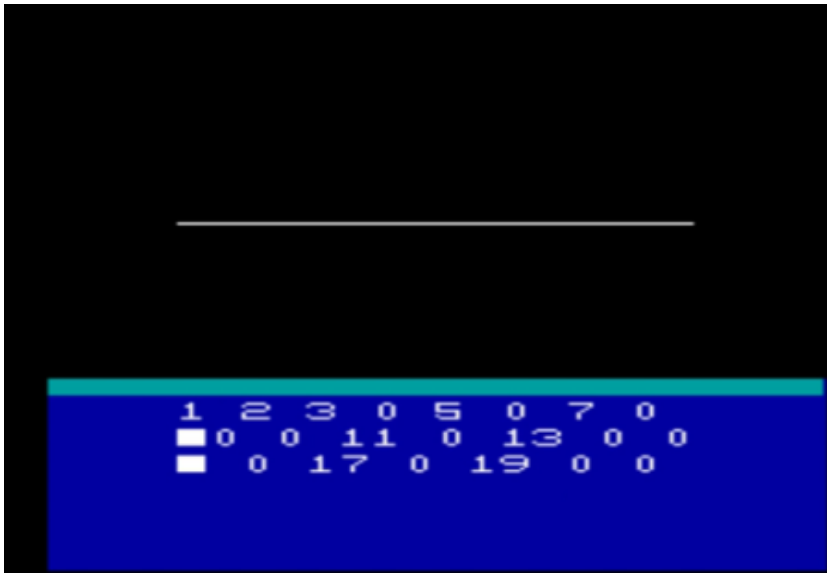
Up for 11. This method is fairly slow. The algorithm requires an increasing amount of time as the numbers being checked grow larger.

Next, consider a different algorithm for finding primes, the Sieve of Eratosthenes.

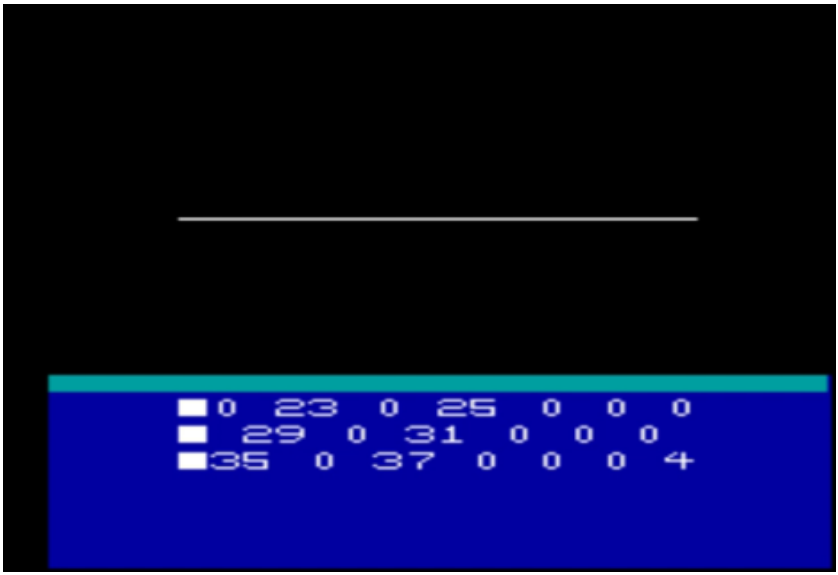
```
PROG SIEVE
N=INDEX 198
I=2
WHIL I LE SIZE N * .5
DO
   IF N(I) EQ 0
   THEN EXIT
   ELSE
      J=I x I
      WHILE J LE SIZE N
      DO
         N(J)=0
         J=J+I
      ENDW
   ENDI
   I=I+1
ENDW
N
ENDP
```

SIZE N
198
S

To run the sieve for numbers up to and including 198, start by creating an array using index 198.

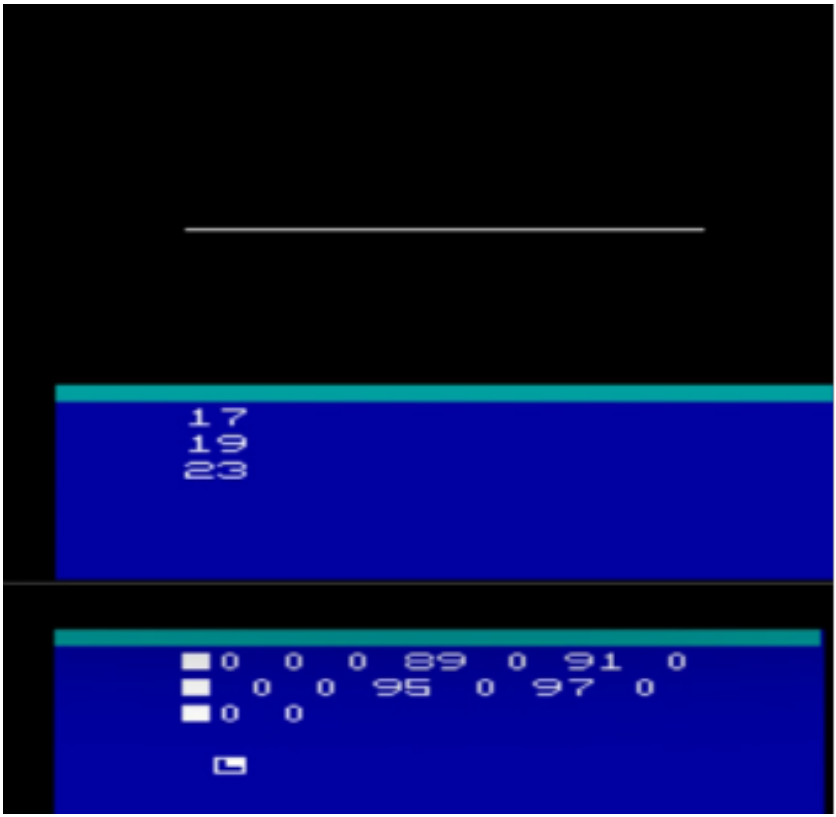


Run the program and the array will be listed, with every number but the primes zeroed out.

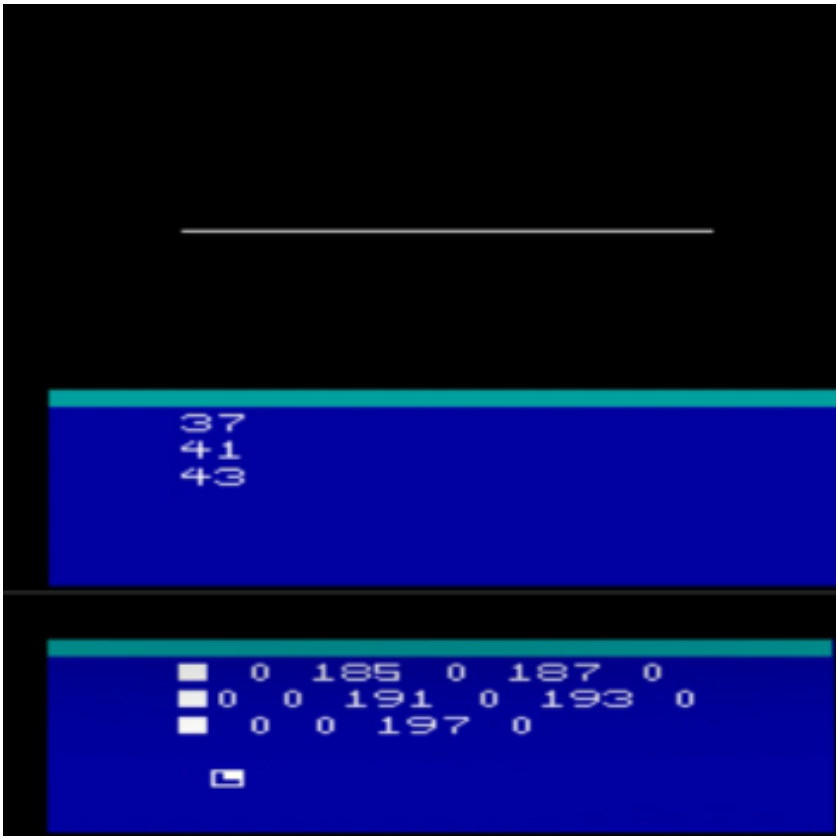

0 23 0 25 0 0 0
29 0 31 0 0 0
35 0 37 0 0 0 4

Only the primes remain when the array is printed.

However, we are constrained by memory, and can only compute primes up to the size of the largest array the system can store.


17
19
23
0 0 0 89 0 91 0
0 0 95 0 97 0
0 0

Comparing the speed of the two algorithms (with the bar chart primaility indicators removed), the sieve method can list all primes under 100 before the slower algorithm can list a prime above 23.


37
41
43
0 185 0 187 0
0 0 191 0 193 0
0 0 197 0

It can print all primes up to 198 before the slower method can compute a prime above 43.

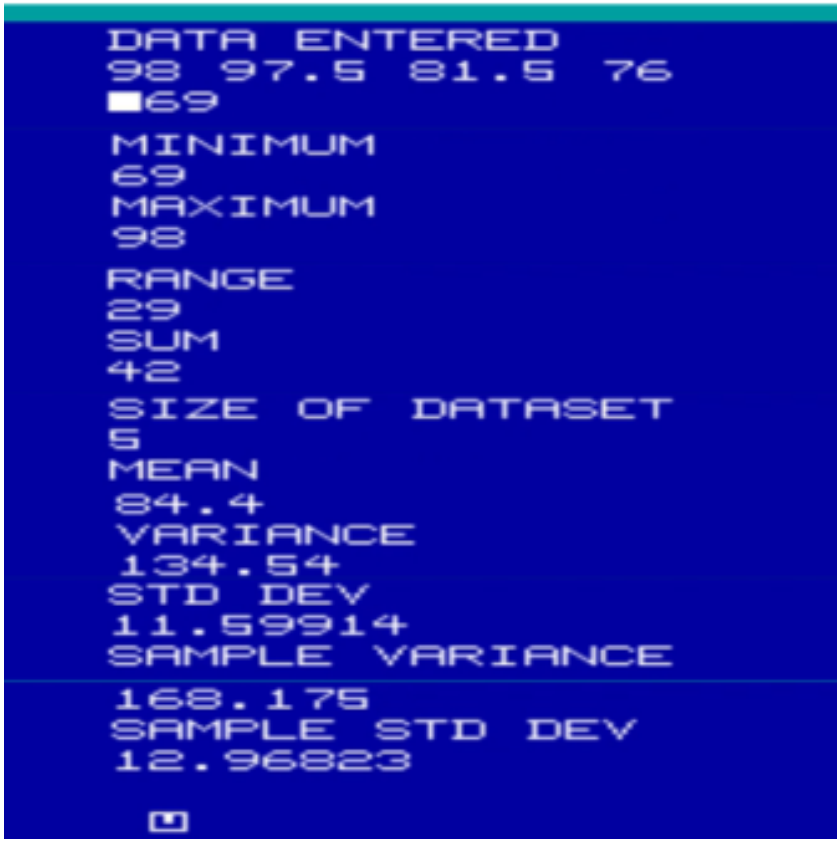
Data Statistics

A program to calculate statistics about a data set. For a given set of numbers, the program will list the minimum, maximum, range and sum of the data, the size of the data set, the mean, variance, standard deviation, sample variance, and sample standard deviation.

```
PROG DSTATS
"ENTER YOUR DATA"
X=KEYB
"DATA ENTERED"
X
"MININUM"
MIN/X
"MAXIMUM"
MAX/X
"RANGE"
MAX/X - MIN/X
"SUM"
+/X
"SIZE OF DATA SET"
N=SIZE X
N
"MEAN"
M=(+/X)÷N
M
"VARIANCE"
V=+/((X-M)*2)÷N
V
"STD DEV"
V*.5
"SAMPLE VARIANCE"
V=+/((X-M)*2)÷(N-1)
V
"SAMPLE STD DEV"
V*.5
ENDP
```

ENTER YOUR DATA
?98 97.5 81.5 76
69

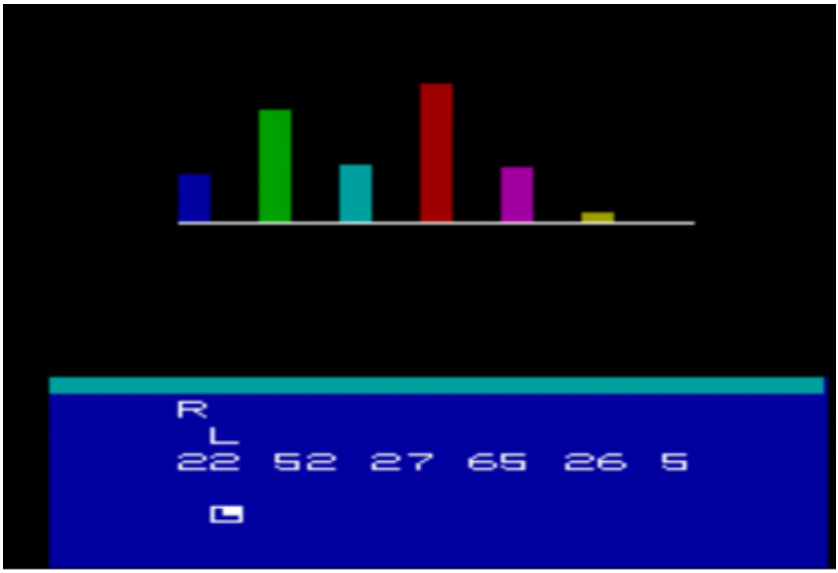
A sample run.


DATA ENTERED
98 97.5 81.5 76
69
MINIMUM
69
MAXIMUM
98
RANGE
29
SUM
42
SIZE OF DATASET
5
MEAN
84.4
VARIANCE
134.54
STD DEV
11.59914
SAMPLE VARIANCE
168.175
SAMPLE STD DEV
12.96823

Lotto Picker

Choose five non-repeating numbers between 1 and 69, and an additional bonus number between 1 and 26. In this algorithm we explicitly model each of the 69 main balls in an array and zero them out as they are selected, checking each time if the selected ball is already zeroed or not.

```
PROG LOTTO

// Pick 5 between 1 and 69
// Pick 1 between 1 and 26

B=INDEX 69
P=RAND 26
W=ARRAY 5
C=1
WHILE C LE 5
DO
   R=RAND 69
   IF B(R) NE  0
   THEN
      W(C)=R
      B(R)=0
      C=C+1
   ENDI
ENDW
W,P
BARC INDEX 6
BARH .9x(W,P)
ENDP
```


R
L
22 52 27 65 26 5

Sample run.

```
PROG LOTTO2

// Pick 5 BETWEEN 1 and 69
// Pick 1 between 1 and 26

P=RAND 26
W=ARRAY 5
C=1
WHILE C LE 5
DO
   R=RAND 69
   IF (+/ (R EQ W) EQ 0)
   THEN
      W(C)=R
      C=C+1
   ENDI
ENDW
W,P
BARC INDEX 6
BARH .9x(W,P)
ENDP
```

A different method uses add reduction instead, requiring less much storage.

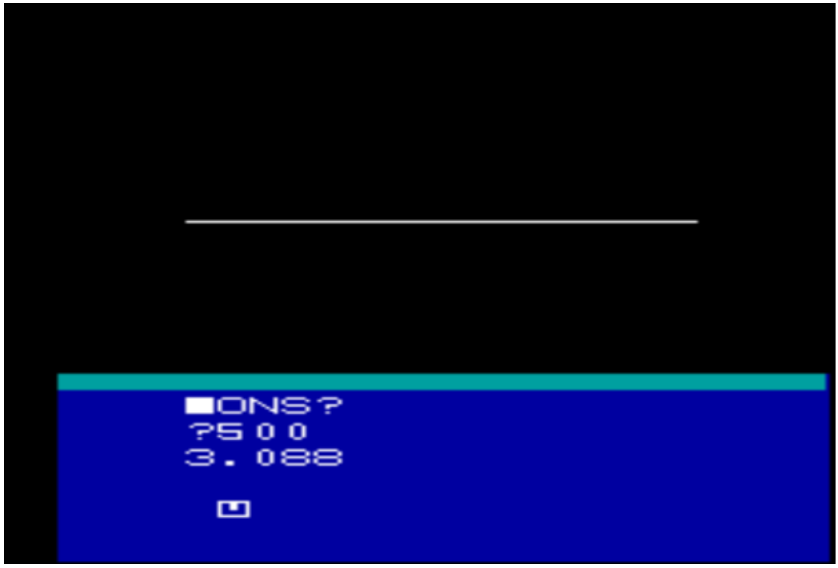
Estimate Pi

This program uses a Monte Carlo algorithm to estimate the value of pi.

```
PROG PI
IN=0
C=1
"HOW MANY ITERATIONS?"
I=KEYB
WHILE C LE I
DO
   X=(RAND 1000000) ÷ 1000000
   Y=(RAND 1000000) ÷ 1000000
   D=(X×X)+(Y×Y)
   IF D LE 1
   THEN IN=IN+1
   ENDI
   C=C+1
ENDW
(4×IN) ÷ I
ENDP
```

PI
HOW MANY ITERATI
■ONS?
?100
■ONS?
?100
2.96
■

PI
HOW MANY ITERATI
ONS?
?500

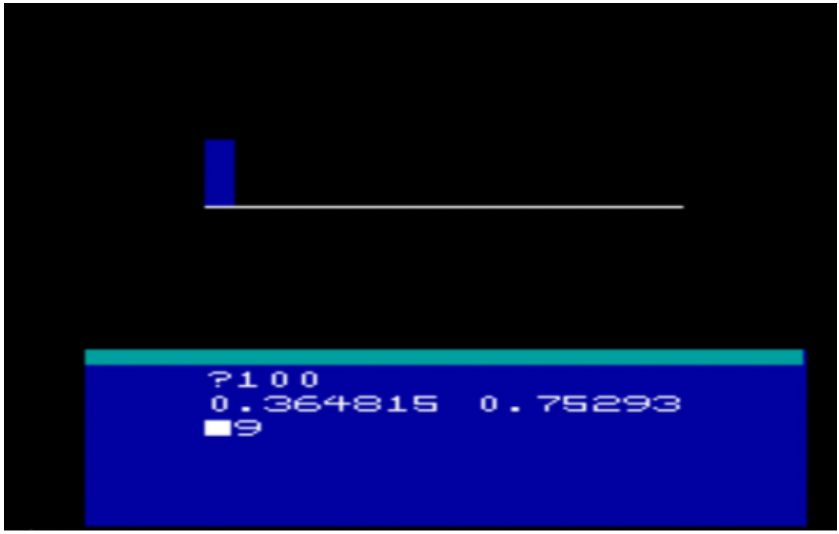

ONS?
?500
3.088

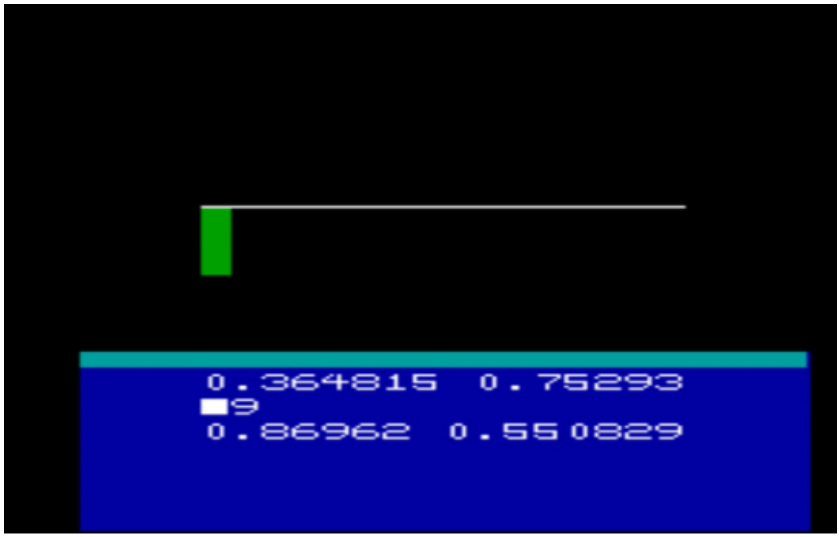
```
PROG PI
IN=0
C=1
"HOW MANY ITERATIONS?"
I=KEYB
WHILE C LE I
DO
   X=(RAND 1000000) ÷ 1000000
   Y=(RAND 1000000) ÷ 1000000
   X,Y
   D=(X×X)+(Y×Y)
   IF D LE 1
   THEN IN=IN+1
      BARC C
      BARH 30
   ELSE
      BARC C
      BARH -30
   ENDI
   C=C+1
ENDW
(4×IN) ÷ I
ENDP
```

We can use the bar chart to show if the random points fall in the unit circle, to visualize the process.

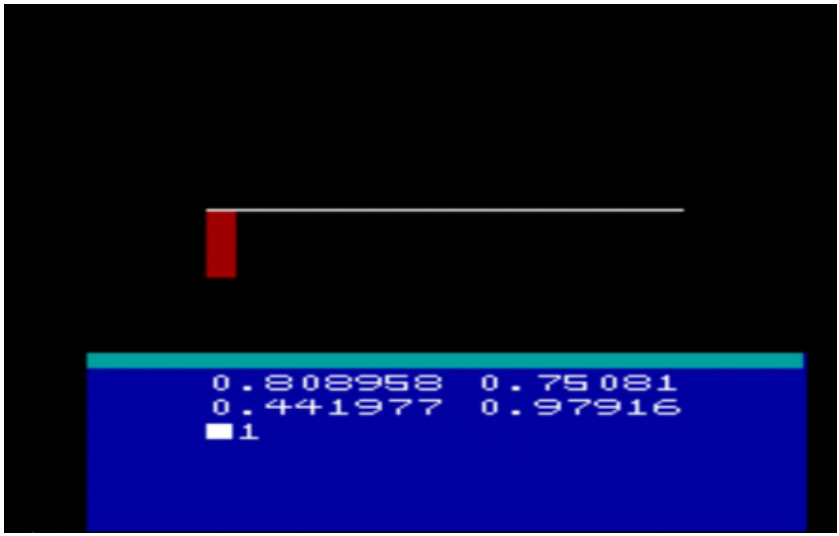
The bar chart value will be positive or negative depending on if the point is in the unit circle.

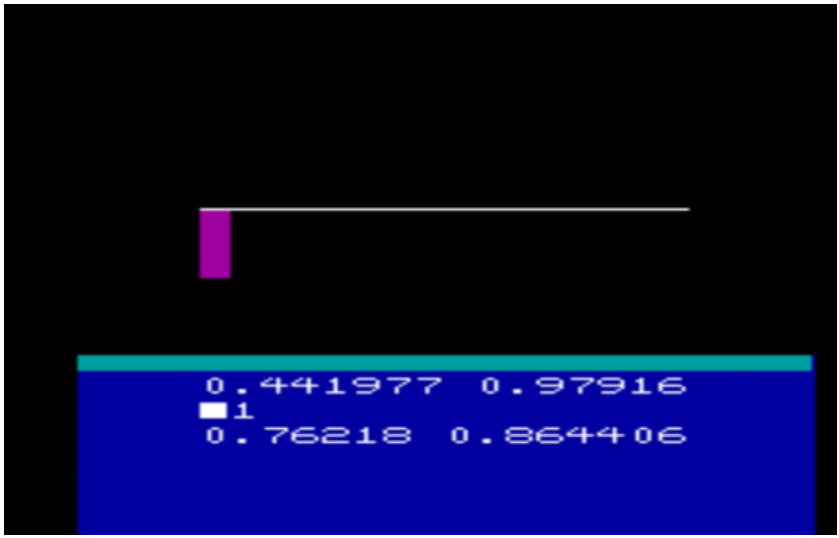
PI
HOW MANY ITERATI
ONS?
?100

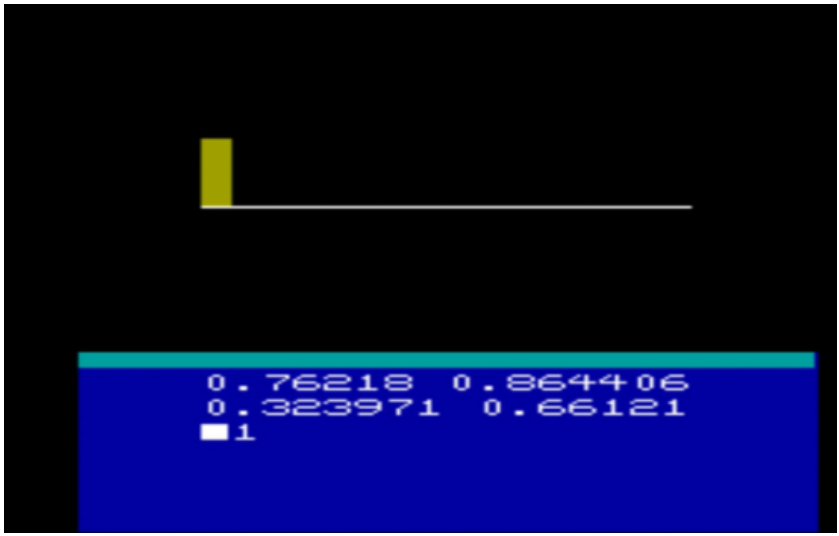

?100
0.364815 0.75293
9


0.364815 0.75293
9
0.86962 0.550829

9
0.86962 0.550829
0.808958 0.75081


0.808958 0.75081
0.441977 0.97916
1


0.441977 0.97916
1
0.76218 0.864406


0.76218 0.864406
0.323971 0.66121
1

1
0.154152 0.38503
3

After 100 points are checked, the estimated value of pi is 3.12.

Taylor, Maclaurin, and Madhava-Leibniz Series

See https://www.mathsisfun.com/algebra/taylor-series.html

Exponential

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$$

Sine

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$

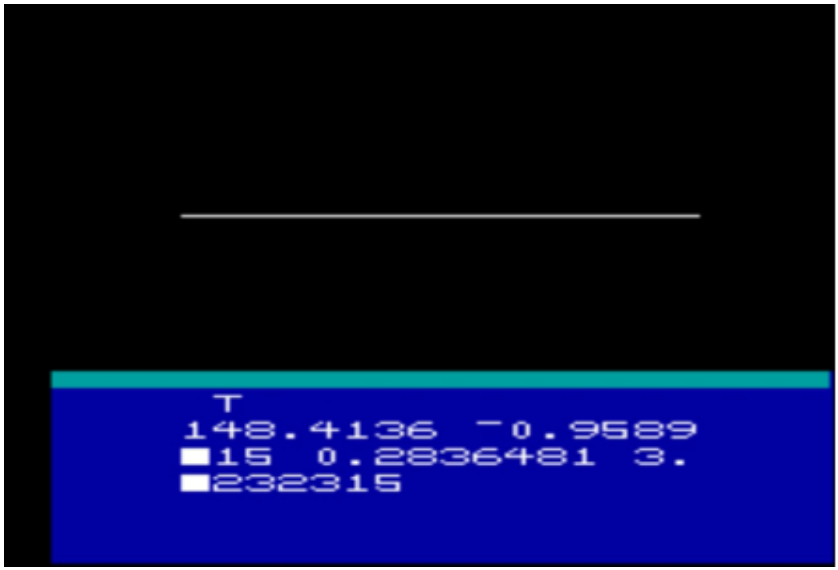
Cosine

$$\cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \dots$$

Arctan (Leibniz Pi)

$$\arctan(1) = \frac{\pi}{4} = 1 - \tfrac{1}{3} + \tfrac{1}{5} - \tfrac{1}{7} + \dots$$

```
PROG TAYLOR
X=5
TICK=1
N=22
I=1
E=1
S=0
C=1
A=0
WHIL I LE N
DO
   E=E+(X*I÷!I)
   IF I MOD 2 EQ 0
   THEN
      TICK=-1×TICK
      C=C+(TICK×(X*I÷!I))
   ELSE
      S=S+TICK×(X*I÷!I)
      A=A+TICK×(1÷I)
   ENDI
   I=I+1
ENDW
E,S,C,(A×4)
ENDP
```


T
148.4136 ¯0.9589
■15 0.2836481 3.
■232315

Running the program for X = 5, extending out to 22 terms. Arctan is multiplied by 4 to get an approximation of pi.

Stag Hunt

In game theory, the stag hunt models a tension between individual safety and mutual cooperation.

Two hunters independently choose between pursuing a large prey that requires cooperation or settling for a smaller one they can obtain alone. Hunting the larger animal succeeds only if both participants commit. If either defects, the attempt fails. By contrast, the smaller prey can always be caught individually, but yields a much lower payoff. Each participant faces a tradeoff between the certainty of a modest reward and the risk of aiming for a larger shared benefit.

```
PROG STAG
WHIL 1
DO
   BARH 0 0 0
   'STAG OR HARE? 1 OR O"
   I=KEYB
   IF I EQ 99
   THEN EXIT
   ENDI
   C=-1+RAND 2
   IF I EQ 1
   THEN
      IF C EQ 1
      THEN
         "I CHOSE STAG TOO"
         Q=10
      ELSE
         "I CHOSE HARE."
         Q=1
      ENDI
   ELSE
      IF C EQ 0
      THEN
         "I CHOSE HARE TOO"
         Q=5
      ELSE
         "I CHOSE STAG."
         Q=1
      ENDI
   ENDI
   BARC 2 3 4
   BARH ((-20+I×40),(-20+C×40),(Q×6))
   'YOUR PAYOUT IS"
   Q
ENDW
ENDP
```

S
STAG OR HARE? 1
OR 0
?1

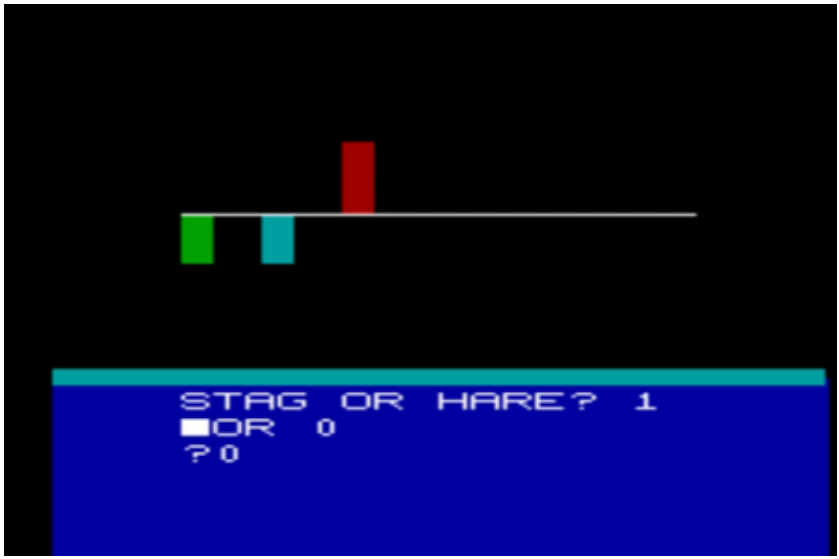
A sample run.

STAG OR HARE? 1
OR 0
?1

?1
I CHOSE HARE.
YOUR PAYOUT IS
1

STAG OR HARE? 1
OR 0
?0
I CHOSE HARE TOO

?0
I CHOSE HARE TOO
YOUR PAYOUT IS
5


STAG OR HARE? 1
OR 0
? 0

?0
I CHOSE STAG.
YOUR PAYOUT IS
1

Moo

Moo, or Bulls and Cows, is a classic code-guessing game. The computer chooses a hidden number and the player attempts to discover it through successive guesses, using feedback from each attempt to narrow the possibilities. The game relies entirely on logical deduction rather than chance. As written, the program reveals the code for demonstration purposes, so remove the line where the code is printed to play the game.

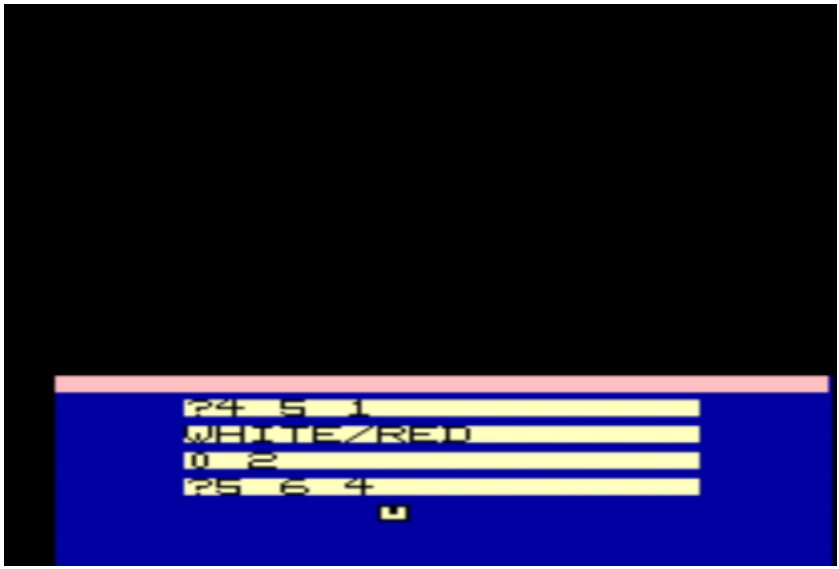
The computer selects a three or four digit number with no repeated digits. After the player guesses, the computer reports how many digits are correct and correctly placed (red “bulls”) and how many are correct but in the wrong position (white “cows”).

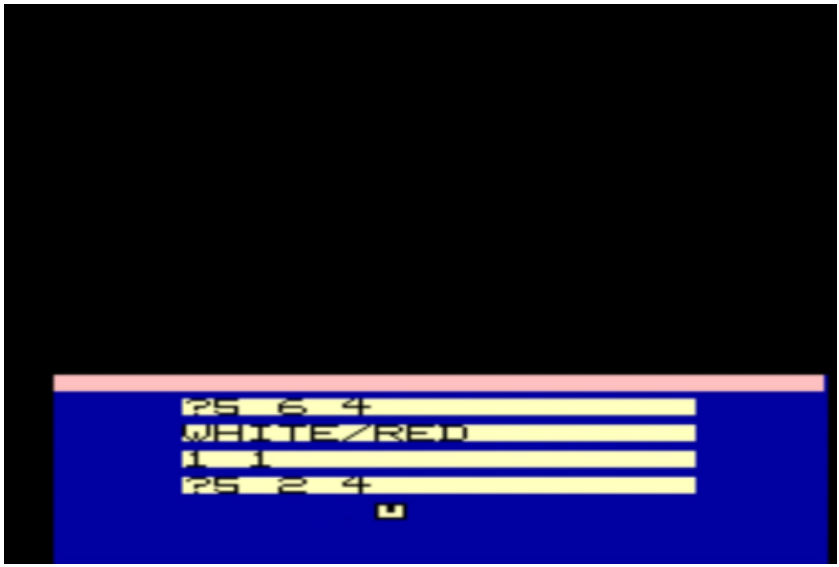
Long before modern commercial adaptations such as *Mastermind*, the game circulated informally and appeared early in computing culture. During the 1970s it was implemented on time-shared mainframe systems, where it was commonly known as MOO. Variants were ported to Multics and early Unix systems, and versions were distributed through user groups and software libraries for DEC machines. Implementations appeared in several languages, including BASIC and FOCAL, and a PDP-8 version later influenced a commercial version of the game released by Milton Bradley.

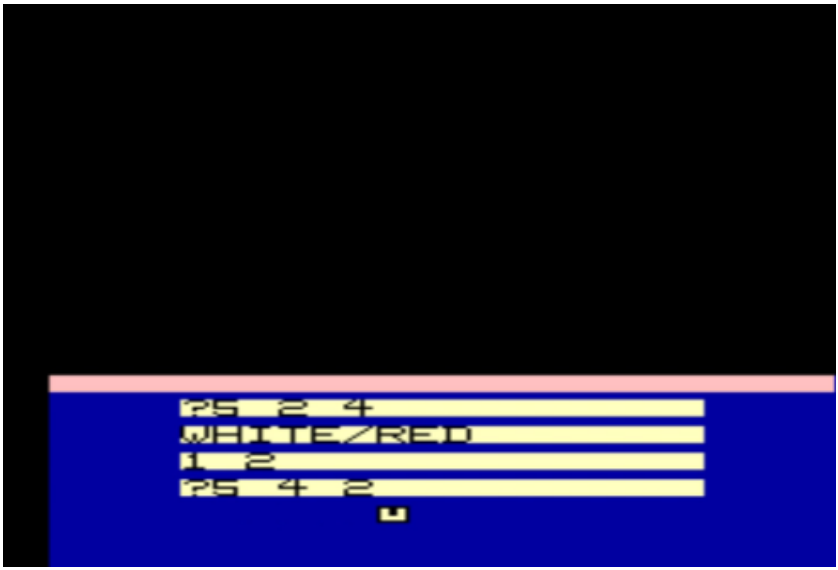
```
PROG MOO
WHIL 1
DO
   Z= RAND 5 5 5
   IF +/(Z(1) EQ Z) EQ 1
   THEN
      IF +/(Z(2) EQ Z) EQ 1
      THEN
        EXIT
      ENDI
   ENDI
ENDW
"GUESS THE 3 NUMBERS"
WHIL 1
DO
   I=KEYB
   IF I(1) EQ 99
   THEN
      EXIT
   ENDI
   X=+/(Z EQ I)
   IF 3 EQ X
   THEN "YOU WIN"
      EXIT
   ELSE
      C=X
      J=1
      R=0
      WHIL J LE 3    //CORRECT COLOR WRONG SPOT
      DO
         IF Z(J) NE I(J)
         THEN
            R=R+ (+/(Z(J) EQ I ) )
         ENDI
      J=J+1
      ENDW
   "WHITE/RED"
   C,R
   ENDI
ENDW
ENDP
```

M
5 4 2
GUESS
?1 4 3

P1 4 3
WHITE/RED
1 0
P4 5 1

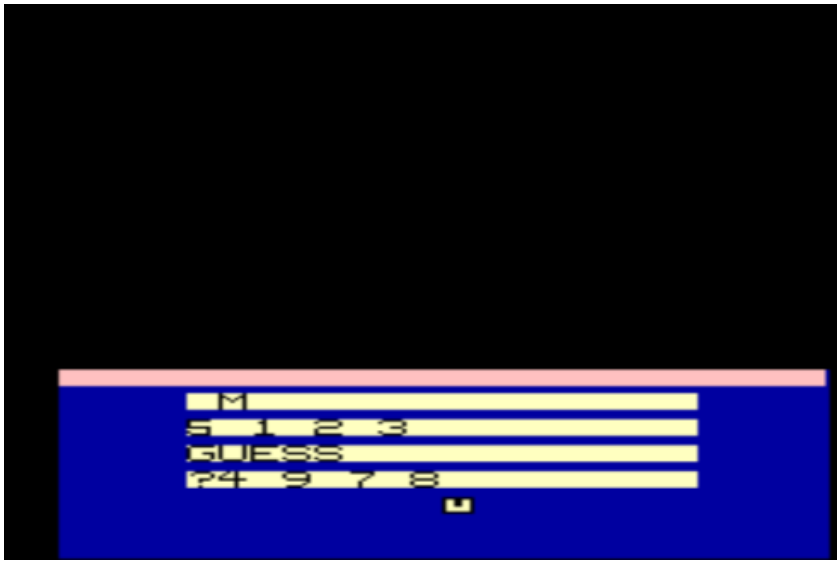

?4 5 1
WHITE/RED
0 2
?5 6 4


25 6 4
WHITE/RED
1 1
25 2 4


25 2 4
WHITE/RED
1 2
25 4 2

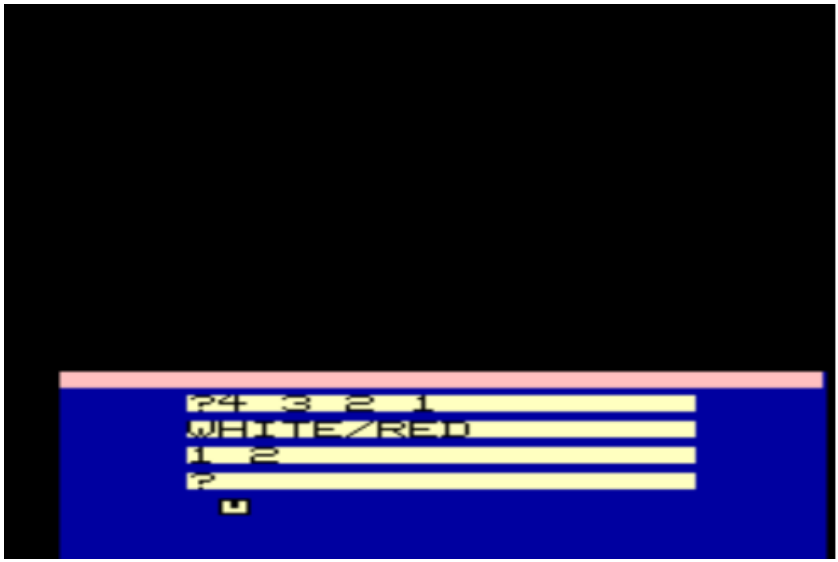
1 2
?5 4 2
YOU WIN

```
PROG MOO4
WHIL 1
DO
   Z= RAND 5 5 5 5
   IF +/(Z(1) EQ Z) EQ 1
   THEN
      IF +/(Z(2) EQ Z) EQ 1
      THEN
         IF +/(Z(3) EQ Z) EQ 1
         THEN
           EXIT
         ENDI
      ENDI
   ENDI
ENDW
"GUESS THE 4 NUMBERS"
WHIL 1
DO
   I=KEYB
   IF I(1) EQ 99
   THEN
      EXIT
   ENDI
   X=+/(Z EQ I)
   IF 4 EQ X
   THEN "YOU WIN"
      EXIT
   ELSE
      C=X
      J=1
      R=0
      WHIL J LE 4    //CORRECT COLOR WRONG SPOT
      DO
         IF Z(J) NE I(J)
         THEN
            R=R+(+/(Z(J) EQ I ))
         ENDI
      J=J+1
      ENDW
   "WHITE/RED"
   C,R
   ENDI
ENDW
ENDP
```


M
5 1 2 3
GUESS
?4 9 7 8

?4 9 7 8
WHITE/RED
0 0
?1 2 3 4

?1 2 3 4
WHITE/RED
0 3
?


?4 3 2 1
WHITE/RED
1 2
?
5

25 3 1 2
WHITE/RED
1 3
?

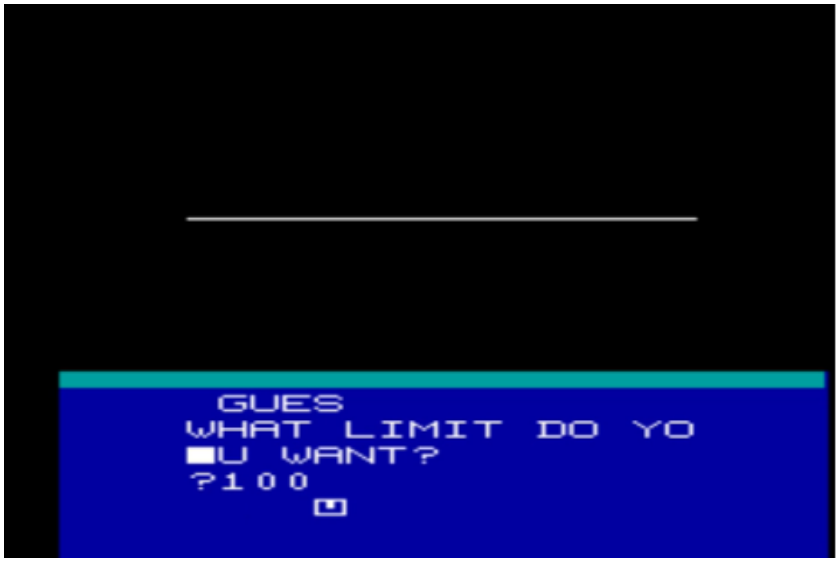
25 2 1 3
WHITE/RED
2 2
?

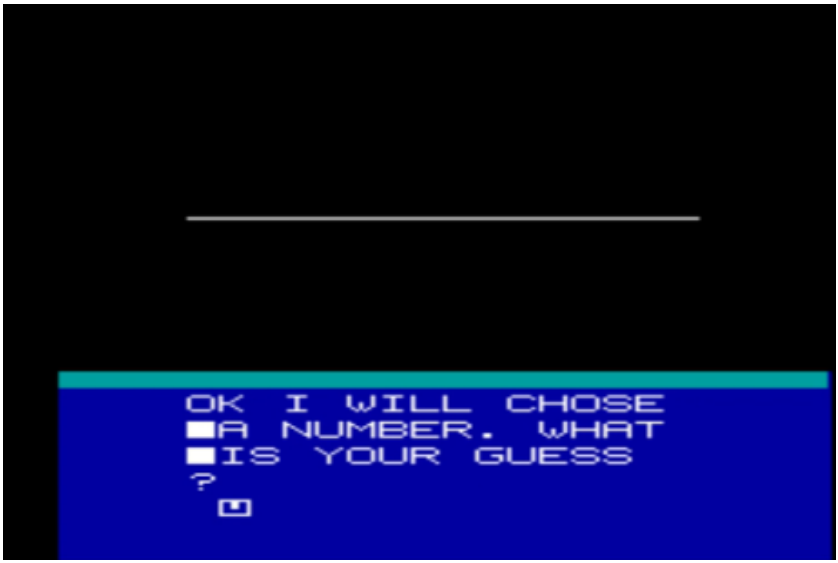
2 2
25 1 2 3
YOU WIN

Ecte's Guess

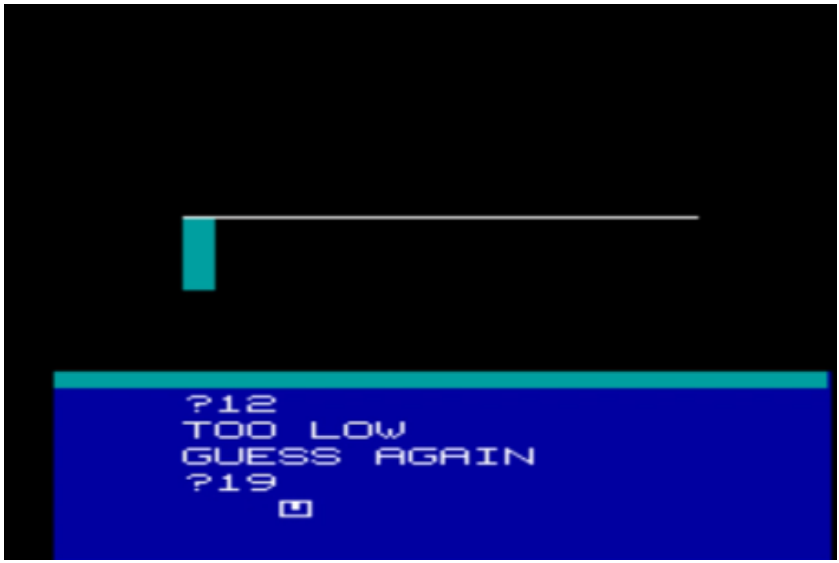
A port of David Ahl's GUESS. Dedicated to my friend Ecte, who enjoys porting this game to new languages.

```
PROG GUES
BARC 3
   “WHAT LIMIT DO YOU WANT”
   LIM=KEYB
   COUNT=1
   NUM=RAND LIM
   “OK I CHOSE A NUMBER. WHAT IS YOUR GUESS”
   WHILE NUM NE G=KEYB
   DO COUNT=COUNT+1
      IF G LT NUM
      THEN “TOO LOW”
          BARH -30
      ELSE “TOO HIGH”
         BARH 30
      ENDIF
      “GUESS AGAIN”
   ENDWHILE
   BARH 0
    “YOUR SCORE VERSUS BINARY SEARCH”
    COUNT, (1+ FLOOR (LOG LIM ÷  LOG 2))
ENDP
```

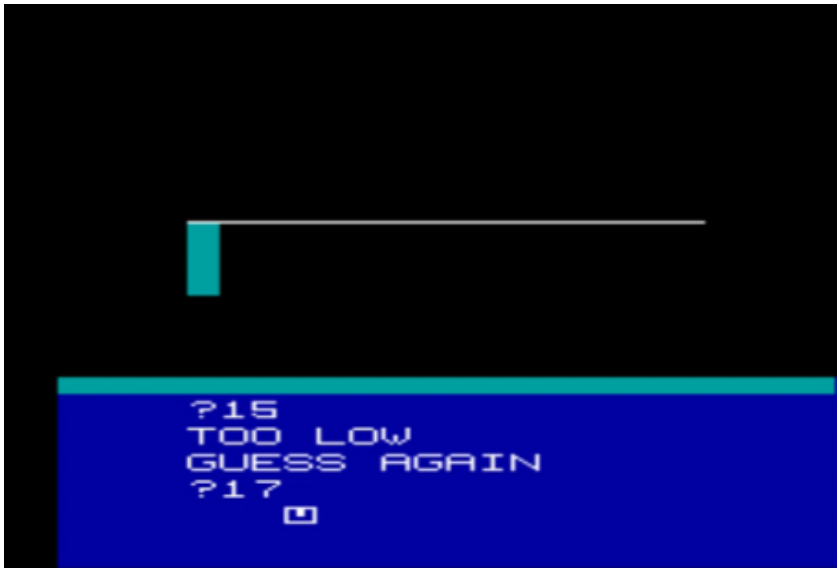

GUES
WHAT LIMIT DO YO
U WANT?
?100


OK I WILL CHOSE
A NUMBER. WHAT
IS YOUR GUESS
?

?26
TOO HIGH
GUESS AGAIN
?12


?12
TOO LOW
GUESS AGAIN
?19

?19
TOO HIGH
GUESS AGAIN
?15


?15
TOO LOW
GUESS AGAIN
?17

?17
TOO HIGH
GUESS AGAIN
?16

?16
YOUR SCORE VERSU
■S BINARY SEARCH
6 7

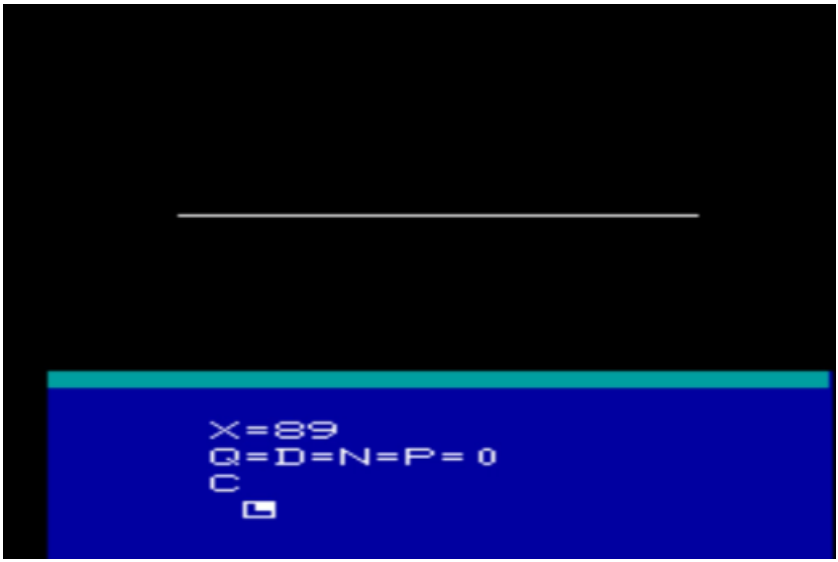
Recursive Coin Change Maker

This program counts out coins to give as change.

Outside the program, set $Q = D = N = P = 0$, and set X to the amount of money to count.

```
Q=D=N=P=0
```

```
Q=D=N=P=0
X=89

PROG C
IF X GE 25
THEN Q=Q+1
   X=X-25
ELSE
   IF X GE 10
   THEN D=D+1
      X=X-10
   ELSE
      IF X GE 5
      THEN N=N+1
         X=X-5
      ELSE
         P=X
         X=0
        ENDI
    ENDI
ENDI
"X Q D N P"
X,Q,D,N,P
IF X GT 0
   THEN C
ENDI
ENDP
```

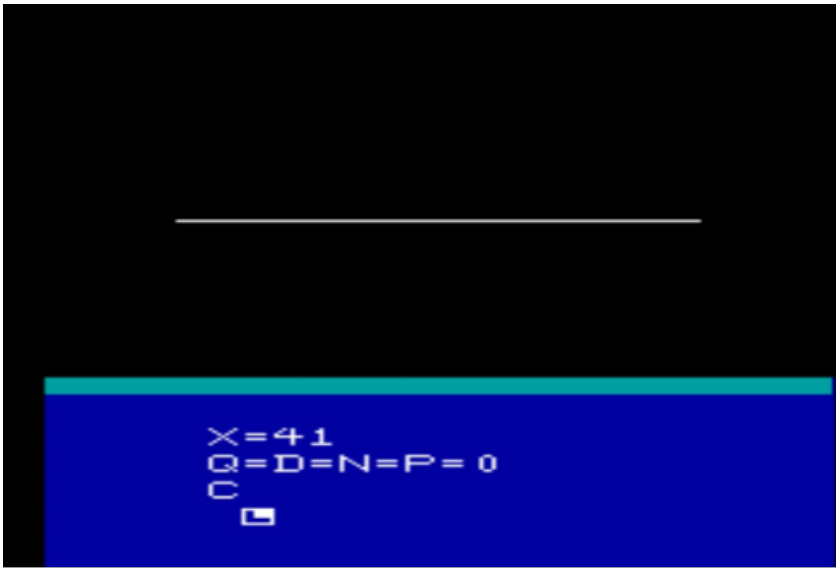

X=89
Q=D=N=P=0
C

C
X Q D N P
64 1 0 0 0

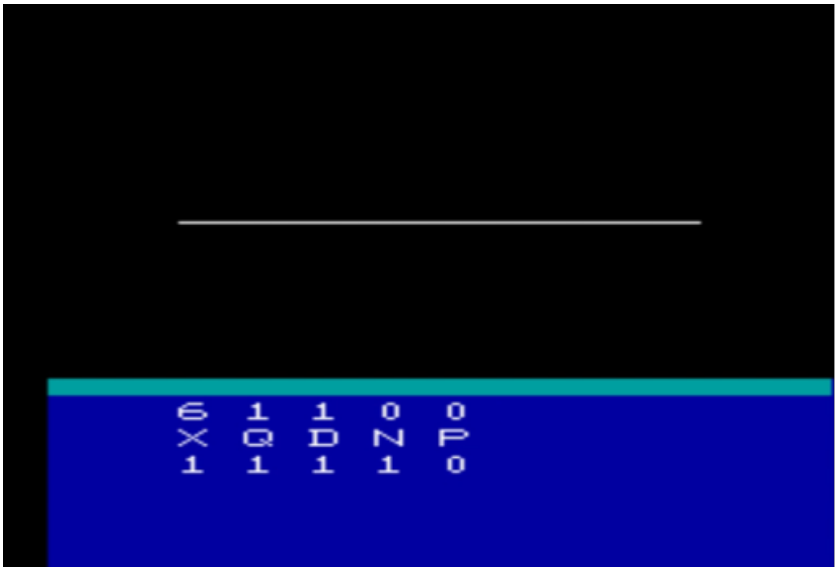
64 1 0 0 0
X Q D N P
39 2 0 0 0

14 3 0 0 0
X Q D N P
4 3 1 0 0

4 3 1 0 0
X Q D N P
0 3 1 0 4


X=41
Q=D=N=P=0
C

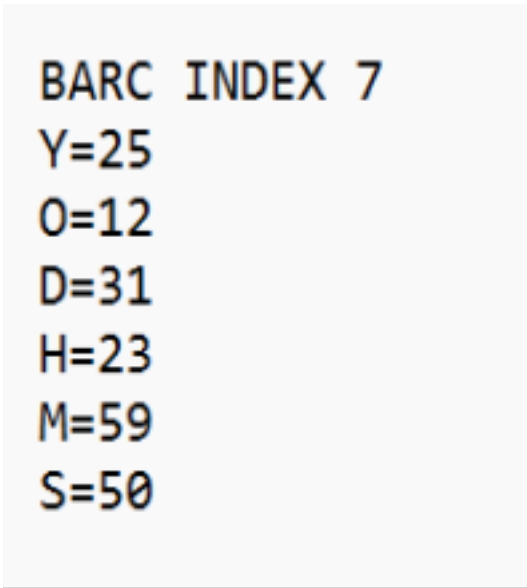
C
X Q D N P
16 1 0 0 0


6 1 1 0 0
X Q D N P
1 1 1 1 0

1 1 1 1 0
X Q D N P
0 1 1 1 1

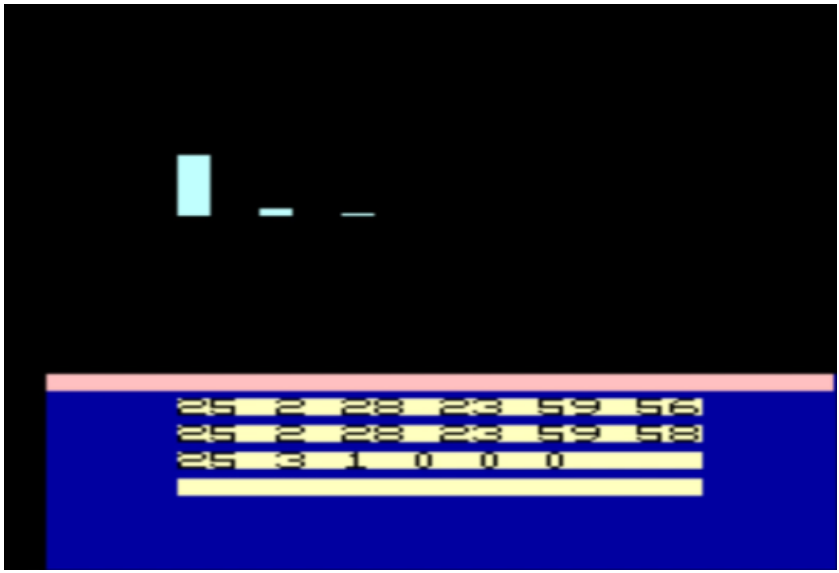
Calendar Clock

Enter in the program, but before running it, first assign values for the year, month, day, hour, minute, and second variables. Set the bar chart colors, too.

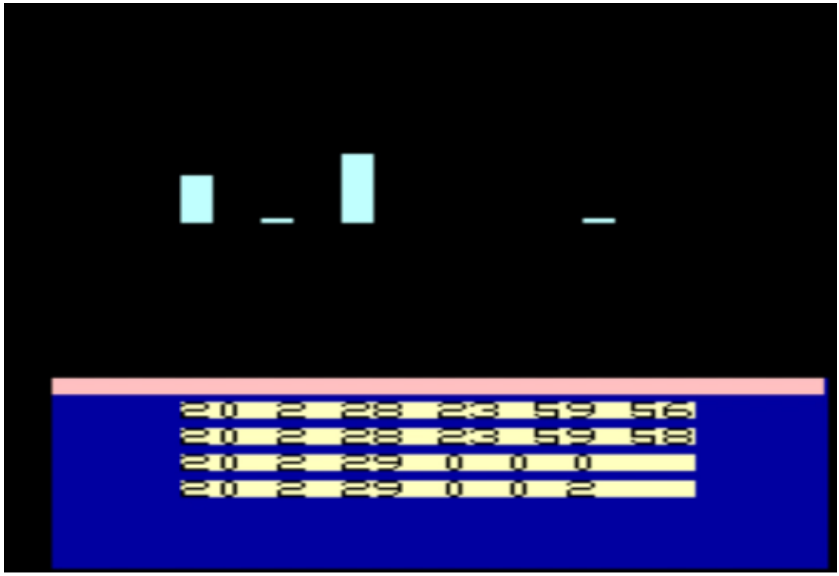


For example, these variables are for New Years Eve 2025, a ten second countdown to 2026.

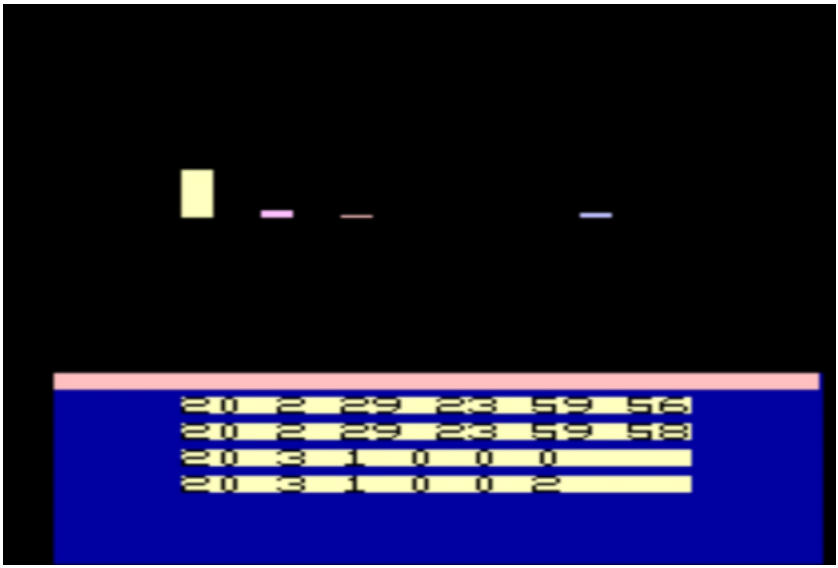
```
PROG CLOCK
WHIL 1
DO
   S=S+2
   IF S GE 60
   THEN S=0
      M=M+1
   ENDI
   IF M EQ 60
      THEN M=0
         H=H+1
   ENDI
   IF H EQ 24
   THEN H=0
      D=D+1
   ENDI
   IF (D EQ 30) AND (O EQ 2)
   THEN D=1
      O=3
   ENDI
   IF (D EQ 29) AND (O EQ 2) AND ((Y MOD 4) NE 0)
   THEN D=1
   O=3
   ENDI
   IF (D EQ 31) AND (+/(O EQ 4 6 8 9 11))
   THEN D=1
      O=O+1
   ENDI
   IF D EQ 32
   THEN D=1
      O=O+1
   ENDI
   IF O EQ 13
   THEN O=1
       Y=Y+1
   ENDI
   BARH Y,O,D,H,M,S
   Y,O,D,H,M,S
ENDW
ENDP
```

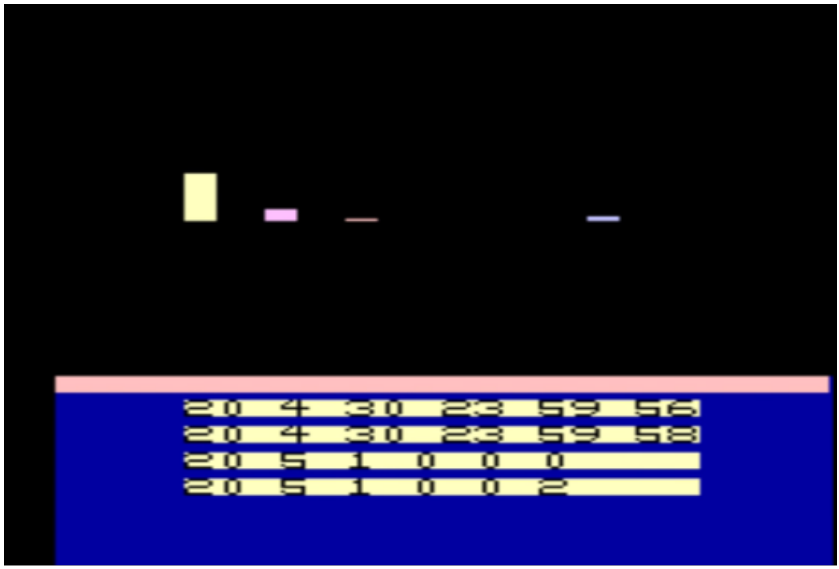


A non-leap year.

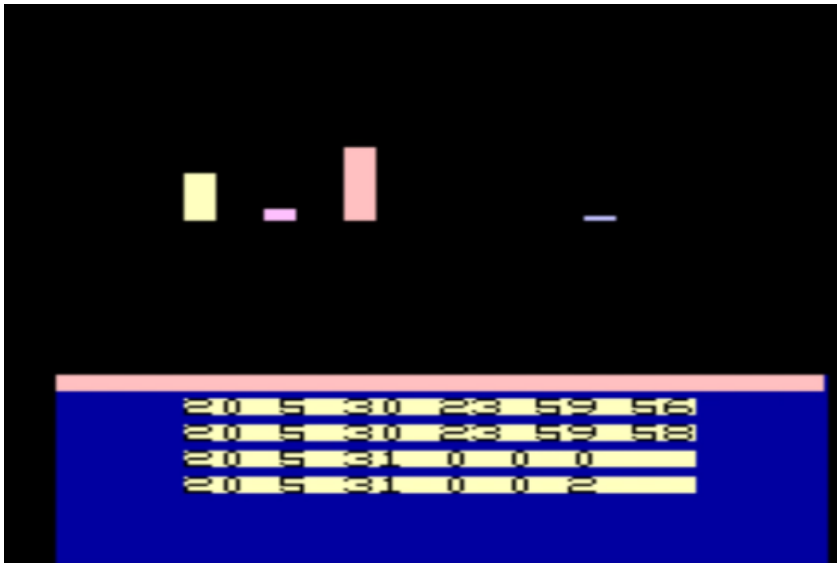


A leap year.

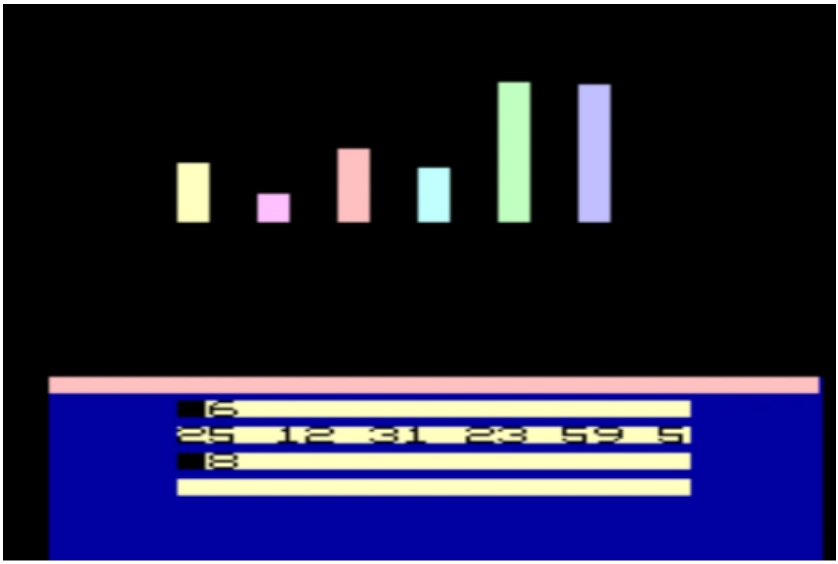

20 2 29 23 59 56
20 2 29 23 59 58
20 3 1 0 0 0
20 3 1 0 0 2

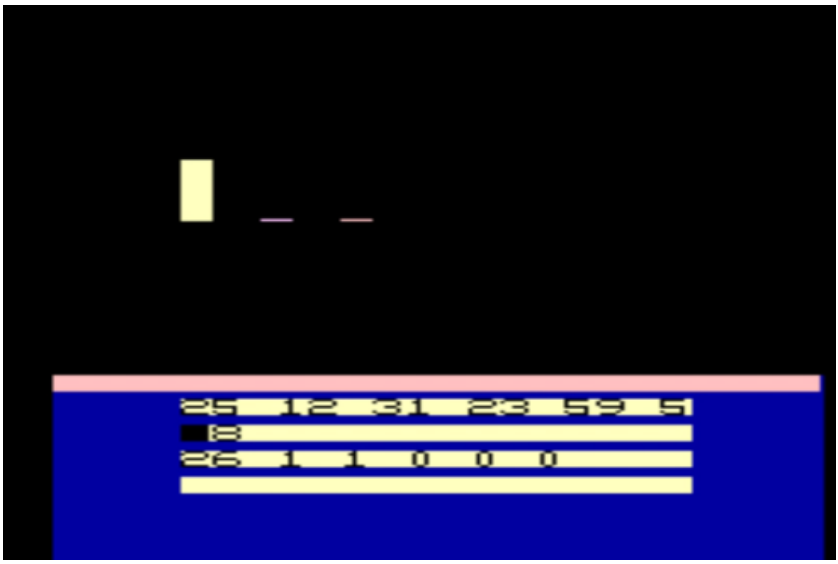


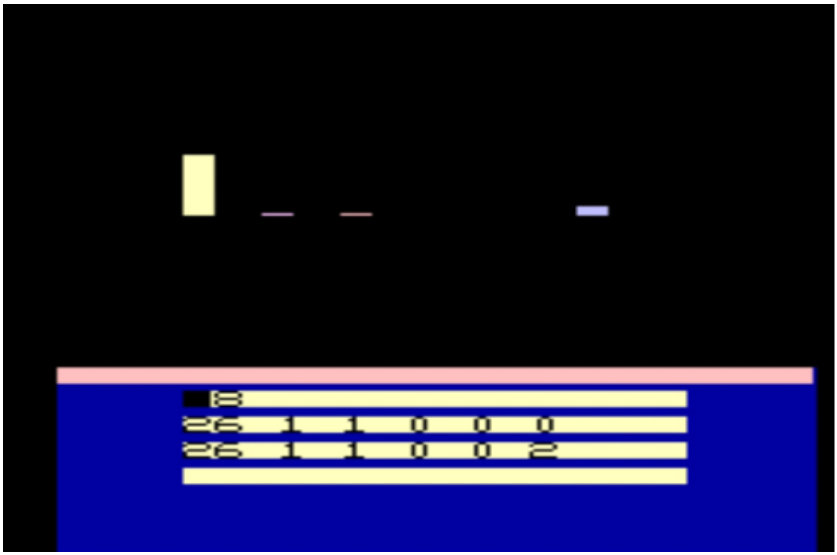
April has 30 days.



May has 31.


6
25 12 31 23 59 5
8


25 12 31 23 59 5
8
26 1 1 0 0 0



Happy New Year!

Credits

All material from works authored or edited by David Ahl is in the public domain.

See https://blog.adafruit.com/2022/06/16/david-ahl-places-all-his-classic-computing-publications-into-the-public-domain/

The APL/S games in this book are collected here:

https://github.com/jamescprice/21-APL-Games-and-One-Liners-For-Your-Umtech-Videobrain/blob/main/21games.zip – All the games in this book.

Preservation copies of the VideoBrain APL/S cartridge ROM are maintained by third-party retrocomputing archives for historical reference and research use. This publication includes APL/S program listings in printed form but does not include ROM images. For discussion see: https://groups.io/g/Channel-F-and-VideoBrain/topic/101298602

Documentation of The Computational Language APL/S:

https://aplwiki.com/wiki/APL/S Great technical description of the language at APL Wiki

https://archive.org/details/byte-magazine-1979-12/page/n81/mode/2up “APL/S: An Alternative” by Robert G. Brown BYTE magazine 12/1979

https://archive.computerhistory.org/resources/access/text/2024/06/102804524-05-001-acc.pdf – Two papers: "An Introduction to APL/S". Conference Proceedings of the Third West Coast Computer Faire” by Robert G. Brown p247 and “VideoBrain and the APL/S Language: First Pass at Everyman’s Computer” by Ted Haynes p246

https://archive.org/details/apls-brochure/APLS-Brochure/ – Includes Lunar Lander game https://forum.vcfed.org/index.php?threads/i-have-been-blessed-with-videobrain-documentation.80042/ - Links to zip file with great documentation and schematics

https://forum.vcfed.org/index.php?threads/videobrain.71462/page-2 – Lots of documentation here!

Audio Discussion:

https://archive.org/details/camvchm_000053/camvchm_000053_t01_a_access.mp3 - ”This is the recording of the session titled ‘High level languages for micros,’ and the following papers were presented: ‘VideoBrain and the APL/S language: First pass at everyman's computer,’ presented by Ted Haynes. ‘An introduction to APL/S: A modern computational language for personal computing,’ presented by Robert G. Brown.”

https://exhibits.stanford.edu/silicongenesis/catalog/nc497jv0418 – An interview with Albert Yu. 00:12:06 Discusses leaving Intel for Video Brain, where he developed microprocessors for use in consumer electronics. 00:17:52 Discusses leaving Video Brain in 1979 to sell home computers between the United States and China.

https://randoc.wordpress.com/category/umtech-videobrain/ - Links to audio of conference talk

https://www.orphanedgames.com/videobrain/

More links:

http://hcvgm.org/Static/Manuals/VideoBrain/VB_Unit_Manual.pdf - System Manual with image of keyboard keys -

https://randoc.wordpress.com/category/umtech-videobrain/ - A good discussion with many links

https://www.atariprotos.com/othersystems/videobrain/cartridges/apls/apls.htm

“The existence of APL/S has been speculated for years. More than one person has claimed to own the cartridge back in the day and it was mentioned in several advertisements, but no actual APL/S cartridge has ever been found. However in 2021 the first evidence of APL/S's existence turned up in the form of a manual in an eBay auction. Unfortunately the cartridge and box were nowhere to be seen. This all changed in 2023 when a complete copy of APL/S was finally located. Only 91 APL/S cartridges were ever made making it the rarest non-prototype VB cartridge.”

http://hcvgm.org/VideoBrain.html - A keyboard diagram, machine specs

http://blog.kevtris.org/blogfiles/videobrain/videobrain_unwrapped.txt – A hardware deep-dive

https://www.oldcomputers.net/videobrain.html – Great images of system internals

https://www.ballyalley.com/articles_and_news/creative_computing_article.pdf – Two Creative Computing articles

Don't miss out!

Click the button below and you can sign up to receive emails whenever James Price publishes a new book. There's no charge and no obligation.

https://books2read.com/r/B-A-GJGCE-SIVNI

Connecting independent readers to independent writers.